
# UNIVERSIDAD CARLOS III DE MADRID

**ESCUELA POLITÉCNICA SUPERIOR**

Ingeniería en Informática

Proyecto Fin de Carrera

# ESTUDIO COMPARATIVO DE PARAMETRIZACIÓN DE UN SISTEMA DE REDES DE NEURONAS ARTIFICIALES EVOLUTIVAS PARA LA PREDICCIÓN DE SERIES TEMPORALES Y HERRAMIENTA SOFTWARE PARA REALIZAR ESA PREDICCIÓN

Autor: D. Miguel Ángel Lobato Álvarez

Tutor: D. Juan Peralta Donate

Enero 2012

Proyecto Fin de Carrera

# ESTUDIO COMPARATIVO DE PARAMETRIZACIÓN DE UN SISTEMA DE REDES DE NEURONAS ARTIFICIALES EVOLUTIVAS PARA LA PREDICCIÓN DE SERIES TEMPORALES Y HERRAMIENTA SOFTWARE PARA REALIZAR ESA PREDICCIÓN


Autor

D. MIGUEL ÁNGEL LOBATO ÁLVAREZ

Tutor

D. JUAN PERALTA DONATE


La defensa del presente Proyecto Fin de Carrera se realizó el día __ de _____ de 2012, siendo evaluada por el siguiente tribunal:

Presidente:

Vocal:

Secretario:

Y habiendo obtenido la siguiente CALIFICACIÓN:

LEGANÉS, a __ de _____ de 20__

# AGRADECIMIENTOS

*Mi más sincero agradecimiento para Juan Peralta ya que sin su insistencia y su apoyo hubiera sido imposible llevar a su fin este proyecto.*

*A mi mujer Mónica y a mis pequeños devoradores de tiempo, Carla (10 meses) y Saúl (2 años), que han sacrificado el tiempo que he dedicado al proyecto y me han animado a transmitirles que hay que culminar las metas que uno se marca en la vida.*

*Y por supuesto a mis padres que, muy a su pesar, me introdujeron de pequeño en el mundo de la informática y sacrificaron mi compañía por darme una buena educación.*

# ÍNDICE DE CONTENIDO

# ÍNDICE DE ILUSTRACIONES

# ÍNDICE DE ECUACIONES

# ÍNDICE DE TABLAS

# ACRÓNIMOS

| Acrónimo | Significado |
|---|---|
| ADANN | Diseño Automático de Redes de Neuronas Artificiales |
| TSBM | Modelos de Predicción Basados en Series Temporales |
| IA | Inteligencia Artificial |
| AG | Algoritmo Genético |
| RNA | Red Neuronal Artificial |
| RNAE | Red Neuronal Artificial Evolutiva |
| SNNS | Stuttgart Neural Network Simulator |
| IPVR | Instituto de Sistemas de Alto Rendimiento Paralelos y Distribuidos |
| RCS | *Rochester Connectionist Simulator* |
| GNU | *GNU Not Unix* |
| GCC | *GNU Collection Compilers* |
| MSE | Error Cuadrático Medio |
| SMAPE | Error Simétrico Porcentual Absoluto. |

# 1 INTRODUCCIÓN

En los Ensayos Históricos de Encarta, que reflejan el conocimiento y la visión de destacados historiadores, Peter N. Stearns, de la Universidad Carnegie Mellon, considera que la capacidad para predecir de forma precisa el futuro depende de cómo se interprete y entienda el pasado.

La humanidad siempre ha estado interesada en predecir el futuro. A lo largo de los años, las sociedades han tratado de adivinar el futuro usando diferentes métodos. Algunos grupos intentaron atisbar lo venidero mediante rituales mágicos o el contacto con lo sobrenatural. Para ello podían leer los augurios en las entrañas de los animales sacrificados o en las hojas de té.

En China, Grecia y Roma clásicas, así como en el Oriente Próximo islámico la astrología fue una "ciencia" esencial. La astrología es el conjunto de creencias que pretende conocer y predecir el destino de las personas mediante la observación de la posición y el movimiento de los astros, y con ese conocimiento pronosticar los sucesos futuros. A pesar de que hoy en día la comunidad científica la considera una pseudociencia, millones de personas creen en ella o la practican.

La mayoría de los pronósticos con los que contamos hoy en día, como los que se relacionan con la política económica o militar, utilizan la historia, debido a que quienes predicen asumen la conexión entre hechos pasados, presentes y futuros.

Existen tres grandes tipos de formas de predicción:

- Ciclos y analogías. Se basa en presunciones sobre la repetición de los hechos y patrones históricos. Este pensamiento subyace tras la conocida frase: "Quienes no conocen el pasado están condenados a repetirlo". Probablemente la primera utilización sistemática del conocimiento histórico para la predicción del futuro asumía que la historia de la humanidad se movía por ciclos, es decir, que lo que

había sucedido antes podía suceder nuevamente después.

- Discontinuidad histórica. En esta la predicción parte de la creencia de que irrumpirá algún tipo de fuerza que cambiará radicalmente el curso de la historia, y por tanto el propio futuro. En este caso el pasado no es una guía para el futuro sino más bien una medida de lo dramático que será el cambio.
- Extrapolación de tendencias. Esta utiliza la historia para identificar tendencias. Aunque la extrapolación de tendencias se ha utilizado siempre hasta cierto punto, ha ganado popularidad en las últimas décadas. Esto es debido a la explosión del conocimiento científico y a la demanda creciente de los gobiernos y los poderes financieros de disponer de los mejores pronósticos posibles de lo que deparará el futuro.

En este proyecto se van a tratar de mejorar las predicciones sobre series temporales.

## *1.1. Series temporales*

Una serie temporal es una secuencia cronológica de observaciones de una variable concreta. Normalmente las observaciones se toman a intervalos regulares ( días, meses, años), pero podrían ser irregulares. Ejemplos típicos de series temporales son la carga de pasajeros de una aerolínea, la tasa de desempleo, el Producto Interior Bruto (PIB) o el cierre diario del índice Dow-Jones. En la figura 1 se puede ver un ejemplo de gráfica de una serie temporal.

El análisis de series temporales comprende métodos para analizar datos de series temporales con el objetivo de extraer estadísticas significativas y otras características de los datos que permitan construir un modelo que represente la serie temporal.

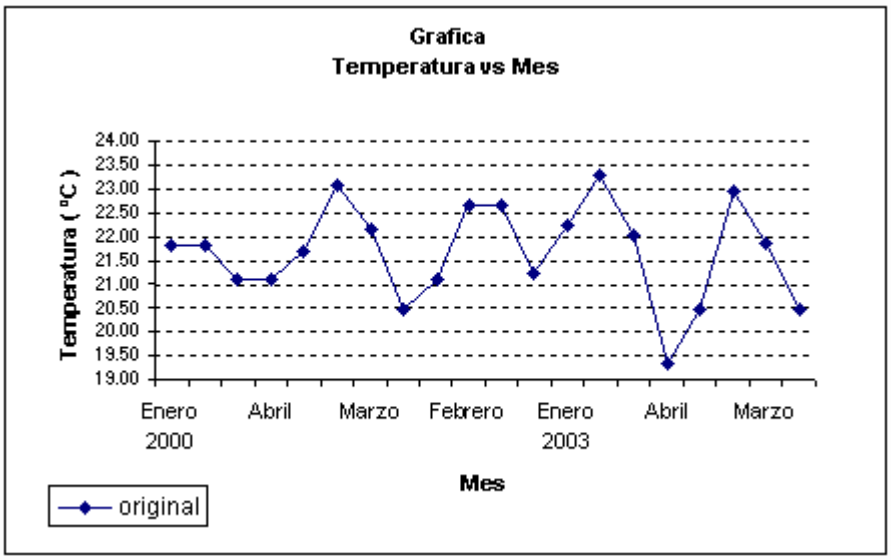


*Figura 1: Ejemplo de gráfica de una serie temporal.*

## *1.2. Predicción de series temporales*

La predicción de series temporales es el uso de un modelo para predecir situaciones futuras basadas en situaciones pasadas ya conocidas, obtener puntos de datos antes de que sean medidos. Un ejemplo de predicción de serie temporal en econometría es la predicción del precio de apertura de una acción bursátil en función de su rendimiento anterior.

La predicción en series temporales es una línea de investigación fundamental en la Estadística. El hecho de poder reproducir el comportamiento de un sistema dinámico a partir de medidas discretas (series temporales) de sus variables posibilita la aplicación de los modelos de predicción basados en series temporales (TSBMs) a innumerables campos del conocimiento, donde los TSBMs complementan la modelización física o, incluso, la sustituyen cuando esta se hace demasiado compleja.

## *1.3. Objetivos del proyecto*

Hoy en día el uso de la inteligencia artificial (IA) y, concretamente de las dos ramas de la misma usadas en este trabajo: los AGs y las RNAs, se puede encontrar en todo lo

que nos rodea en el día a día. Así se pueden ver sistemas que organizan la mercancía que llega a la empresa optimizando su espacio de almacenamiento, sistemas que reconocen la huella dactilar al fichar a la entrada y salida de una oficina, aplicaciones que reconocen las caras de personas en las fotos que subes a Internet, y se podría seguir así enumerando un largo número de aplicaciones. Con esto no cabe duda de que merece la pena ya de por si conocer un poco más en profundidad algo que puede ser aplicado en casi cualquier campo que se toque.

Si esto es interesante mucho más lo es el participar en la mejora de un sistema que ayude a "predecir el futuro" o que, al menos, lo intente con mayor o menor acierto. Muchas empresas están interesadas en conocer con cierta antelación ciertas variables que influyen directamente en su labor empresarial como por ejemplo la masificación de viajeros en aerolíneas, los consumos de ciertos alimentos, etc, esto les ayudaría a gestionar mejor los recursos necesarios para el desarrollo de su actividad.

Siendo más objetivo, el propósito principal a conseguir sería el de intentar optimizar o mejorar en tiempo y/o en precisión los resultados obtenidos con el modelo ADANN propuesto en [1]. Para ello se han planteado varios experimentos que ayuden a lograrlo:

- Variar el número de individuos de la población, se lanzarán dos experimentos uno con 20 individuos y otro con 50. Con esto se podrá ver si al aumentar o disminuir la población el método logra mejores resultados en su predicción o si al menos los mantiene con la población reducida con lo que, seguramente, disminuirá el tiempo de computación.
- Aumentar el número de generaciones de 100 a 250, esperando que al aumentar la población se logre llegar a una predicción más precisa aunque pueda penalizar en el tiempo de computo.
- Variar el método de selección del AG, tratando de observar si la solución llega antes a un resultado esperado o si el resultado es más preciso. Aquí se ha optado por dos variantes a comparar frente al método de torneo de dos padres:
    - Torneo de tres padres: Los sucesores se eligen de entre tres parejas de

padres seleccionados al azar de entre toda la población.

- Ruleta: En este método se simula una ruleta con porciones proporcionales a la aptitud de cada individuo. Se hace girar la ruleta, mediante un valor aleatorio, y se escoge aquel individuo cuya porción este dentro del valor aleatorio generado. Con este método se espera que aquellos individuos con mejor aptitud sean elegidos más veces.

También se ha buscado, con el desarrollo de la utilidad "GetForecastedValues", facilitar la labor de los investigadores, tanto en este proyecto y en las investigaciones actuales que se están llevando a cabo, como en las futuras, ayudando a la extracción y análisis de los resultados obtenidos en los experimentos lanzados que hasta ahora resultaba ser una tarea manual y muy tediosa.

Como objetivo personal esta el llegar al final de un camino que comencé a andar hace ya unos años y que, como otros muchos, dejé abandonado por motivos laborales y a expensas de presentar el proyecto final de carrera. Como dice el dicho: "*más vale tarde que nunca*".

A lo largo de los siguientes capítulos se podrá ver como se encuentra el estado del arte de redes de neuronas artificiales (RNA), algoritmos genéticos (AG), predicción de series temporales (capítulo 2) así como el modelo ADANN sobre el que se han aplicado los experimentos. Además se comentarán las herramientas usadas en el proyecto y como se ha llevado a cabo este mediante una correcta gestión (capítulo 3). En la parte principal de esta memoria (capítulo 4) se muestra un estudio más detallado de los cambios efectuados sobre el sistema para poder lanzar los experimentos, junto a la herramienta desarrollada para la predicción y extracción de resultados. La explicación de los experimentos y los resultados obtenidos se pueden ver en el capítulo 5. Al final de la memoria se analizan las conclusiones obtenidas y las futuras líneas sobre las que dirigir nuevas investigaciones (capítulo 7).

# 2 ESTADO DEL ARTE

La IA es la ciencia e ingeniería de hacer máquinas inteligentes, especialmente aplicaciones inteligentes. Se asemeja a la tarea de usar los ordenadores para entender la inteligencia humana, pero la inteligencia artificial no tiene que limitarse a métodos que son observables biologicamente. Se entiende inteligencia como la parte computacional de la habilidad de alcanzar metas en el mundo.

A continuación se verán los métodos estadísticos más usados para la predicción de series temporales así como las diferentes técnicas de inteligencia artificial usadas en el modelo ADANN para el diseño automático de RNA. Con las modificaciones efectuadas en algunas de ellas se han llevado a cabo los experimentos de este proyecto para demostrar si existen mejoras en la aplicación de los mismos.

## *2.1. Métodos estadísticos*

El método estadístico de predicción de series temporales, y dado que la estadística trata con números, consiste en obtener unos resultados a partir de unos datos numéricos siguiendo determinadas reglas y operaciones. El método sigue los siguientes pasos:

1. Recuento, relevamiento o recopilación de datos. La etapa inicial consiste en la recolección de datos referidos a la situación que se desea investigar. Estos datos brindan información sobre las características de los individuos pertenecientes a la población objeto de estudio.
2. Tabulación y agrupamiento de datos. Gráficos. Los datos recopilados son convenientemente ordenados, clasificados y tabulados, es decir, dispuestos en tablas que facilitan la lectura. Los gráficos permiten una interpretación simple y rápida de los hechos y además, pueden conducir a la elección de los métodos más adecuados para el análisis de los datos.

*3.* Medición de datos. En esta etapa, comienza la elaboración matemática y medición de los datos. Se observa que los datos tienden a centrarse en torno de ciertos valores llamados parámetros o medidas de posición (promedio, mediana, modo). Luego se analiza la dispersión de los datos con respecto a esos valores centrales, se definen entonces los parámetros o medidas de dispersión (desvíos, desviación estándar).

*4.* Inferencia estadística. Predicción. Después de la medición de datos, la Teoría de la Probabilidad acude en ayuda de la Estadística. Se deducen las leyes de inferencia que permiten predecir el comportamiento futuro de la población investigada. De esta manera, la Estadística contribuye al mejoramiento del estado de una población.

Entre los métodos estadísticos usados para la predicción se pueden destacar los siguientes:

- Análisis de regresión múltiple: Se utiliza cuando dos o más factores independientes están implicados en el pronóstico a medio plazo. Se utiliza para evaluar los factores que debe incluir y excluir. Puede ser utilizado para desarrollar modelos alternativos con factores diferentes.
- Regresión no lineal: No asume una relación lineal entre las variables. Se utiliza mucho cuando el tiempo es la variable independiente.
- Análisis de Tendencias: Usa la regresión lineal y no lineal con el tiempo como variable explicativa. Utilizada en patrones con tiempo.
- Análisis de descomposición: Se utiliza para identificar varios patrones que aparecen simultáneamente en una serie temporal. También se utiliza para desestacionalizar una serie, evitando que siempre tome valores en las mismas zonas temporales.
- Análisis de media móvil: Media móviles simples. Predice valores futuros basándose en la media ponderada de valores pasados. Es fácil de actualizar.
- Promedios móviles ponderados: Muy potente y económico. Son ampliamente

utilizados donde se necesitan pronósticos repetidos. Usa métodos como la suma de los dígitos y los métodos de ajuste de tendencia.

- Filtrado adaptativo: Un tipo de media móvil que incluye un método de aprendizaje de errores del pasado. Puede responder a los cambios según la importancia de la tendencia, de manera estacional y según factores aleatorios.
- Suavizado exponencial: Una forma de media móvil de la predicción de series temporales. Eficiente para usar con patrones estacionales. Fácil de ajustar según errores del pasado y preparar predicciones. Ideal para situaciones en las que se necesitan preparar muchas predicciones. Se usan diferentes formas dependiendo de la presencia de tendencias o variaciones cíclicas.
- Filtro Hodrick-Prescott: Es un mecanismo de suavizado usado para obtener un componente de tendencia a largo plazo en una serie de tiempo. Es una manera de descomponer una serie dada en componentes estacionarios y no estacionarios, de tal manera que la suma de los cuadrados de la serie de los componentes no estacionarios sea mínima, con una penalización a los cambios sobre los derivados del componente no estacionario.

## *2.1. Redes de neuronas artificiales*

Las redes de neuronas artificiales nacieron después de que McCulloc y Pitts presentarán un conjunto de neuronas simplificado en 1943. Estas neuronas se representaban como modelos de redes biológicas en componentes de circuitos que podían realizar tareas computacionales. El modelo básico de una neurona artificial se basa en la funcionalidad de la neurona biológica.

Por definición, una neurona es una célula del sistema nervioso que genera y transmite los impulsos nerviosos. Las neuronas están conectadas entre ellas formando circuitos neuronales. En los puntos de contacto hay una pequeñísima separación (la sinapsis) por la que se intercambian unas moléculas químicas llamadas neurotransmisores. Las neuronas tienen unas ramificaciones llamadas dendritas, por donde reciben los impulsos nerviosos de otras neuronas. Cuando la activación recibida es suficiente, la

neurona descarga un impulso nervioso que se trasmite por su axón. El axón es la prolongación larga que transmite la excitación neuronal y que permite enviar la información nerviosa a otras neuronas.

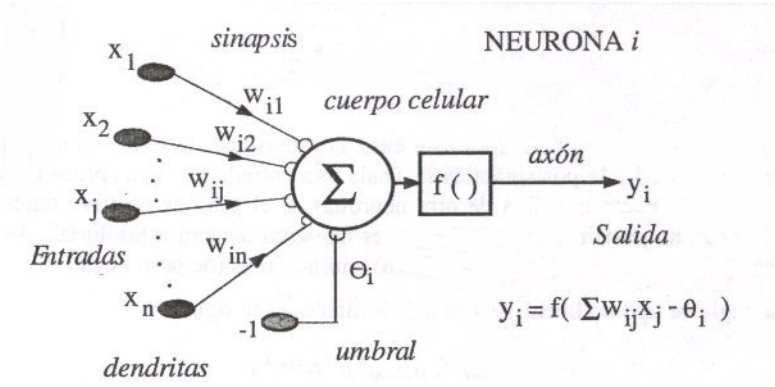


*Figura 2: Red neuronal artificial.*

Las redes de neuronas artificiales son un paradigma de aprendizaje y procesamiento automático inspirado en la forma en que funciona el sistema nervioso de los animales. Se trata de un sistema de interconexión de neuronas en una red que colabora para producir un estímulo de salida.

Los componentes principales de una red neuronal artificial son los siguientes:

- Factores de peso: Normalmente una neurona recibe varias entradas simultaneas. Cada entrada tiene su propio peso relativo que da a la entrada el impacto necesario para el procesamiento de la función suma de los elementos. Los pesos son los coeficientes de adaptación dentro de la red que determinan la intensidad de la señal de entrada según lo registrado por la neurona artificial. Los pesos pueden ser modificados en respuesta a varios conjuntos de entrenamiento y de acuerdo a la topología específica de una red o a través de sus reglas de aprendizaje.
- Función suma: El primer paso de una neurona es calcular la suma ponderada de todas sus entradas. Matemáticamente, las entradas y los pesos correspondientes son vectores. La señal de entrada total es el vector producto de estos dos vectores.

Algunas funciones suma tienen un proceso adicional aplicado al resultado y anterior a la función de transferencia. A este proceso se le denomina función de activación. Su propósito es permitir a la salida de la suma variar con respecto al tiempo. La mayoría de las redes usan una función de activación ,"identidad", que equivale a no tener ninguna.

- Función de transferencia: El resultado de la función suma, casi siempre la suma ponderada, se transforma en una salida a través de un proceso conocido como la función de transferencia. En esta la suma se compara con algún umbral para determinar la salida de la neurona. Si la suma es mayor que el valor del umbral , el elemento procesado genera una señal, si es menor, no hay señal. Ambos tipos de respuesta son significativos.
- Escalado y límite: Tras la función de transferencia, el resultado puede someterse a procesos de escalado y límite. Este escalado simplemente multiplica un factor de escala al valor de transferencia y luego le añade un desplazamiento. El límite es el mecanismo que asegura que el resultado del escalado no supera los límites, tanto el superior como el inferior.
- Función de salida: Cada neurona puede producir una señal de salida que se propaga a cientos de otras neuronas. Normalmente esta salida corresponde al resultado de la función de transferencia. Algunas topologías de red permiten la competición entre neuronas. Esta competición determina que neurona se activará o dará una salida y, en las entradas competitivas, ayudarán a determinar que neurona participará en el proceso de aprendizaje.
- Función de error y valor retropropagado: En la mayoría de las redes se calcula la diferencia entre la salida y la salida deseada. Este error se transforma mediante la función de error para adaptarse a una determinada arquitectura de red. La arquitectura más básica usa el error directamente. Este error se propaga a la función de aprendizaje de otra neurona y se suele denominar el error actual. El error actual normalmente se retropropaga a una capa anterior.

- Función de aprendizaje: El propósito de esta función es modificar los pesos de las entradas de cada elemento de acuerdo a algún algoritmo neuronal. A este proceso también se le conoce por la función de adaptación. Existen dos tipos de aprendizaje:
  - Supervisado: Este requiere de un conjunto de datos de entrenamiento que califique los resultados obtenidos.
  - No supervisado: Cuando no existen resultados en los que basarse para evaluar los resultados obtenidos, el sistema debe de disponer de un criterio interno de la red que la vaya ajustando según las salidas que vaya obteniendo.

Las RNAs, con su habilidad para averiguar datos significativos entre otros complicados o imprecisos, se pueden usar para extraer patrones y determinar tendencias que son demasiado complejas para ser vistas por humanos u otras técnicas computacionales.

Otras ventajas son:

- Aprendizaje adaptativo: Habilidad para aprender como hacer tareas basándose en datos dados para el entrenamiento.
- Auto-organización: Una RNA puede crear su propia organización o representación de la información que recibe durante el tiempo de aprendizaje.
- Tiempo real: Los cálculos efectuados por una red neuronal artificial pueden realizarse en paralelo, por lo que pueden diseñarse y fabricarse dispositivos que tengan en cuenta esta posibilidad.
- Tolerancia a fallos: Esto se consigue a través de codificar información redundante. La destrucción parcial de una red provoca la correspondiente degradación del rendimiento. Sin embargo, algunas capacidades de la red pueden mantenerse incluso con un daño mayor.

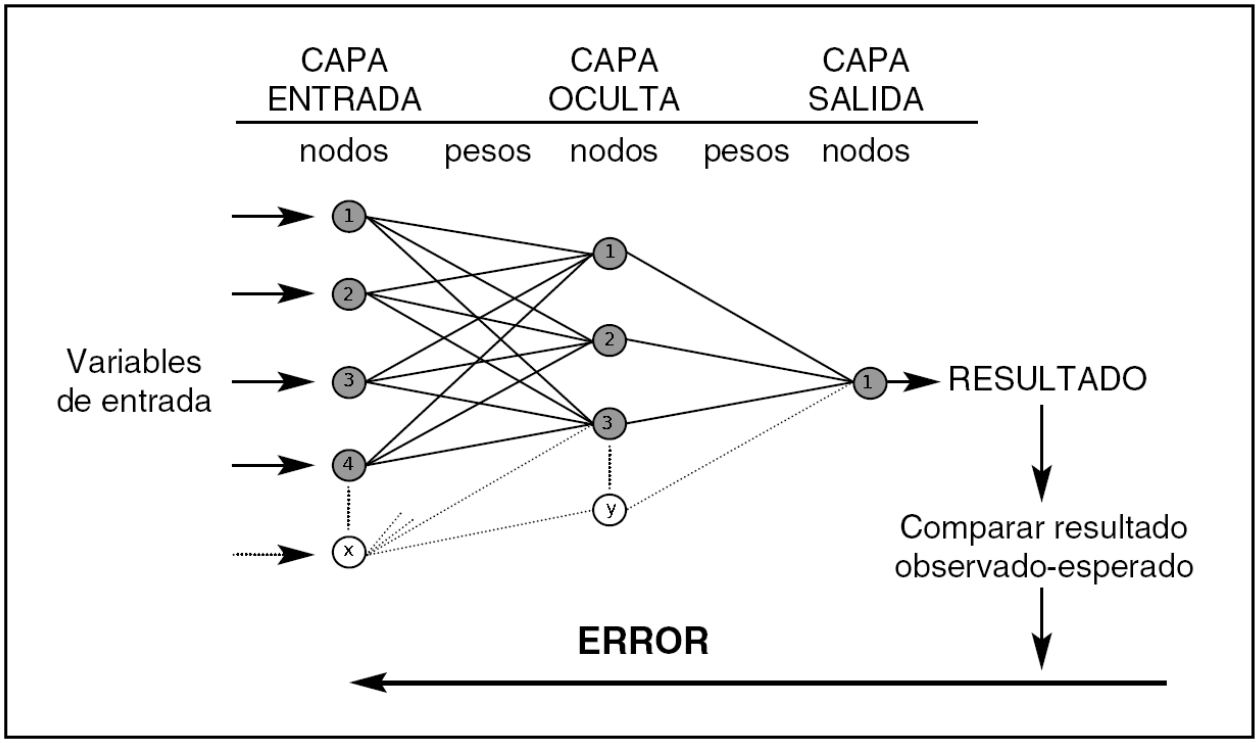


*Figura 3: Red neuronal artificial perceptrón multicapa.*

El perceptrón multicapa es una RNA pro-alimentada con una o más capas entre las capa de entrada y la de salida. Pro-alimentada quiere decir que los datos fluyen en una dirección desde la capa de entrada a la de salida. Este tipo de redes se entrenan con el algoritmo de aprendizaje de retropropagación. Las redes perceptrón multicapa se usan ampliamente en la clasificación de patrones, reconocimiento, predicción y aproximación. Pueden resolver problemas que no son separables linealmente.

En la figura 3 puede observarse un ejemplo de RNA perceptrón multicapa.

La utilidad de los modelos de RNAs cae en el hecho de que pueden usarse para inferir una función desde la observación y también para usarla. Las redes neuronales no supervisadas se pueden usar para aprender representaciones de la entrada que capturen las características más sobresalientes de su distribución, por ejemplo la máquina de Boltzman, y más recientemente algoritmos de aprendizaje profundo, que pueden aprender implícitamente la función de distribución de los datos observados. El aprendizaje en las

RNAs es particularmente útil en aplicaciones donde la complejidad de los datos o procesos hace que el diseño de su función sea impracticable.

Los campos en los que las RNAs se aplican tienden a clasificarse en las siguientes categorías:

- Aproximación de función o análisis de regresión, incluyendo predicción de series temporales y modelado.
- Clasificación, incluyendo reconocimiento de secuencias y patrones, detección y toma de decisiones.
- Procesamiento de datos, incluyendo filtrado, agrupación, separación y compresión.

Las áreas de aplicación de las redes neuronales incluyen sistemas de identificación y control (control de vehículos, de procesos), juegos y toma de decisiones (backgammon, ajedrez, carreras), reconocimiento de patrones (radares, identificación facial, reconocimiento de objetos), reconocimiento de secuencias (gestos, habla, reconocimiento de escritura), diagnosis médicos, aplicaciones financieras, explotación de datos, visualización y filtrado de correo basura.

## *2.2. Algoritmos genéticos*

Los algoritmos genéticos (AGs) son algoritmos adaptativos de búsqueda heurística basados en las revolucionarias ideas de la selección natural y la genética. El concepto básico de los AGs es que son diseñados para simular procesos en un sistema natural necesario para la evolución, específicamente aquellos que siguen los principios establecidos por Charles Darwin de la supervivencia del más apto. Como tales, representan una explotación inteligente de una búsqueda aleatoria dentro de un espacio de búsqueda definido para resolver un problema.

Los AGs surgieron de la mano de John Holland de la Universidad de Michigan. Este comenzó su trabajo en AGs a principios de los 60. Uno de sus primeros logros fue la publicación de "*Adaptation in Natural and Artificial System*" en 1975.

Holland tenía un doble objetivo: mejorar el conocimiento sobre el proceso de adaptación natural y diseñar sistemas artificiales que tuvieran propiedades similares a los sistemas naturales.

La idea básica era la siguiente: la reserva genética de una población dada contiene potencialmente la solución, o una solución mejor, a un problema adaptativo dado. Esta solución no es "activa" porque la combinación genética en la que se basa se divide entre varios sujetos. Sólo la asociación de diferentes genomas puede llevar a la solución. Ningún sujeto tiene el genoma que de la solución, sino que durante la reproducción y el cruzamiento, se produce una nueva combinación genética y, finalmente, un sujeto puede heredar un "gen bueno" de ambos padres que lleve a la solución.

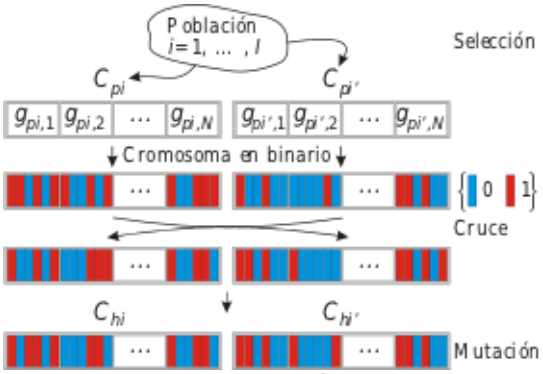


*Figura 4: Reproducción en los AG.*

El método de Holland es especialmente efectivo ya que no sólo considera el papel de la mutación (las mutaciones rara vez mejoran los algoritmos), sino que también usa la recombinación genética (cruzamiento): esta recombinación, el cruzamiento de soluciones parciales mejora enormemente la capacidad del algoritmo para aproximarse al óptimo y a veces encontrarlo.

Los AGs establecen una analogía entre el conjunto de soluciones de un problema, llamado fenotipo, y el conjunto de individuos de una población natural, codificando la información de cada solución en una cadena, generalmente binaria, llamada cromosoma. Los símbolos que forman la cadena son llamados genes. Cuando la representación de los

cromosomas se hace con cadenas de dígitos binarios se le conoce como genotipo. Los cromosomas evolucionan a través de iteraciones, llamadas generaciones. En cada generación, los cromosomas son evaluados usando alguna medida de aptitud. Las siguientes generaciones (nuevos cromosomas), llamada descendencia, se forman utilizando dos operadores genéticos, de cruzamiento y de mutación.

A continuación se explica punto por punto el proceso que sigue un AG básico:

- Inicialización: Se genera aleatoriamente la población inicial, que está constituida por un conjunto de cromosomas los cuales representan las posibles soluciones del problema.
- Evaluación: A cada uno de los cromosomas de esta población se aplicará la función de aptitud para saber cómo es de "buena" la solución que se está codificando.
- Condición de término: El AG se deberá detener cuando se alcance la solución óptima, pero ésta generalmente se desconoce, por lo que se deben utilizar otros criterios de detención. Normalmente se usan dos criterios: correr el AG un número máximo de iteraciones (generaciones) o detenerlo cuando no haya cambios en la población. Mientras no se cumpla la condición de término se hace lo siguiente:
  - Selección: Después de saber la aptitud de cada cromosoma se procede a elegir los cromosomas que serán cruzados en la siguiente generación. Los cromosomas con mejor aptitud tienen mayor probabilidad de ser seleccionados.
  - Sobrecruzamiento: El cruzamiento es el principal operador genético, representa la reproducción sexual, opera sobre dos cromosomas a la vez para generar dos descendientes donde se combinan las características de ambos cromosomas padres.
  - Mutación: Modifica al azar parte del cromosoma de los individuos, y permite alcanzar zonas del espacio de búsqueda que no estaban cubiertas por los individuos de la población actual.

- Reemplazo: una vez aplicados los operadores genéticos, se seleccionan los mejores individuos para conformar la población de la siguiente generación.

Entre las ventajas que demuestran los AGs tenemos las siguientes:

- Los AGs son intrínsecamente paralelos. La mayoría de los algoritmos son en serie y sólo pueden explorar el espacio de soluciones hacia una solución en una dirección al mismo tiempo, y si la solución que descubren resulta subóptima, no se puede hacer otra cosa que abandonar todo el trabajo hecho y empezar de nuevo. Sin embargo, ya que los AGs tienen descendencia múltiple, pueden explorar el espacio de soluciones en múltiples direcciones a la vez. El algoritmo genético puede dirigirse hacia el espacio con los individuos más aptos y encontrar el mejor de ese grupo.
- Resuelve problemas con múltiples soluciones.
- Los AGs se desenvuelven bien en problemas con un paisaje adaptativo complejo, aquéllos en los que la función de aptitud es discontinua, ruidosa, cambia con el tiempo, o tiene muchos óptimos locales
- Los AGs no saben nada de los problemas que deben resolver.
- El AG es un método muy sencillo de entender y que prácticamente no requiere de conocimientos matemáticos.
- Los AGs se pueden adaptar fácilmente a simulaciones y modelos ya existentes.

Pero no todo son ventajas también se pueden comentar algunas desventajas:

- Ciertos problemas de optimización (llamados problemas de variante) no se pueden resolver por medio de AGs.
- No hay una garantía absoluta de que un AG encontrará un óptimo global. Ocurre con frecuencia en poblaciones con muchos individuos.
- Como otras técnicas de IA, el AG no puede garantizar la optimización constante del tiempo de respuesta. Es más, la diferencia entre el tiempo de respuesta óptimo

mayor y el menor puede ser mucho mayor que con los métodos de gradiente convencionales. Esto limita el uso de AGs en aplicaciones de tiempo real.

- Las aplicaciones de los AGs en controles que se realizan en tiempo real están limitadas debido a las soluciones aleatorias y a la convergencia, esto quiere decir que la población entera mejora, pero esto no se puede decir de un individuo en concreto.

## *2.3. Redes de neuronas artificiales evolutivas*

Las redes neuronales artificiales evolutivas (RNAE) pueden ser definidas como la conjunción entre las RNA y los procesos de búsqueda evolutivos. Se pueden tener tres tipos diferentes de evolución en las RNAE, la evolución de los pesos de las conexiones, la evolución de las arquitecturas y la evolución de las reglas de aprendizaje.

### 2.3.1. Evolución de los pesos de la conexión

El proceso de aprendizaje en las RNA puede ser supervisado y no supervisado. El aprendizaje supervisado se basa en el entrenamiento de los pesos de las conexiones para lograr obtener un peso lo más óptimo posible siguiendo algún criterio lógico. Uno de los métodos de aprendizaje supervisado es el algoritmo de retropropagación. Este es un algoritmo de búsqueda de gradiente descendiente que trata de minimizar el error cuadrático medio total entre la salida actual y la salida del objetivo de la RNA. Al ser un algoritmo de búsqueda se puede entender como la evolución de los pesos de las conexiones llevada a cabo por un AG ya que los AG son buenos tratando con espacios de búsqueda largos y complejos. Para conseguir esto es necesario codificar los pesos de las conexiones en el cromosoma del AG y definir como llevar a cabo la evolución del AG. Se consideran dos formas de codificación la binaria y la decimal. La inclusión del AG en las RNA implica un coste computacional más elevado que con el algoritmo de retropropagación, por lo que se puede tener en cuenta métodos combinados que acerquen mediante el AG a una solución parcial y dentro de esta usar el algoritmo de retropropagación para afinar la búsqueda.

### 2.3.2. Evolución de la arquitectura

Hasta ahora el diseño de la arquitectura era una tarea de expertos que la realizaban mediante la técnica de prueba y error. Algunos trabajos, buscando la automatización de este diseño de RNA, se han basado en algoritmos constructivos y destructivos. Los primeros comienzan con una red mínima y van añadiendo nodos y conexiones, si se necesitan, durante el entrenamiento, mientras que los segundos, parten de una red máxima y van eliminando nodos y conexiones sobrantes durante el entrenamiento.

Según Miller et al. [6] el diseño de la evolución basado en un AG posee algunas peculiaridades que lo hacen más apropiado para la búsqueda de este diseño por los siguientes motivos:

- La superficie de búsqueda es inmensa ya que se pueden tener infinitos nodos y conexiones.
- La superficie no es diferenciable ya que los cambios en nodos y conexiones son discretos y pueden afectar positivamente o negativamente al rendimiento de la RNAE.
- La superficie es complicada y con mucho ruido ya que depende de las condiciones iniciales.
- Las RNAE con diseños similares pueden tener habilidades de procesamiento y rendimiento diferentes, esto hace que la superficie no sea fiable.
- RNAE con diferentes arquitecturas pueden tener capacidades similares por lo que la superficie es multimodal.

Para hacer uso de este diseño automático es necesario, al igual que con la evolución de los pesos, codificar la información de la arquitectura en el AG, para lo cual hay que ver cuanta información hay que codificar en el cromosoma. En este punto se tienen dos opciones la codificación directa donde la información de la arquitectura se representa mediante cadenas binarias, es decir, cada nodo y conexión aparecen directamente en la codificación, o codificación indirecta donde sólo se codifican los

parámetros más importantes de la arquitectura.

### *Esquema de codificación directa para RNAE*

Como ya se ha comentado en este esquema de codificación se expresan directamente cada nodo y cada conexión mediante su representación binaria o decimal.

La manera de codificar esta información es utilizando una matriz de *NxN* elementos donde $C=(c_{[ij]})_{NxN}$ representa la conectividad de una RNAE con N nodos y $c_{ij}$ señala si existe una conexión entre el nodo i y el nodo j o no. La cadena que se codifica en el cromosoma es la concatenación de las filas o columnas de la matriz.

Debido al enorme tamaño que pueden alcanzar RNAE grandes, este método no escala muy bien y sólo es válido para redes pequeñas aunque se puede optimizar un poco haciendo uso del conocimiento previo del dominio del problema para restringir el espacio de búsqueda.

Se puede definir fácilmente una función de adaptación y se ha demostrado que, si esta hace uso de del error de test en vez del de entrenamiento, generalizan mucho mejor.

### *Esquema de codificación indirecta para RNAE*

Tal y como ya se explico previamente de forma breve, este esquema de codificación representa tan sólo las características más importantes de la arquitectura en vez de cada nodo y cada conexión. Esto consigue compactar la información de la conectividad de las RNAE. Hay varias formas de llevar a cabo esta codificación, a continuación se van a ver fugazmente alguna de ellas.

- Codificación de los parámetros de conectividad

  Existen varios conjuntos de parámetros para definir la conectividad, por ejemplo las "*blueprints*", que es una representación binaria indirecta compuesta por uno o más segmentos que codifican la conectividad de las RNAE y los parámetros de aprendizaje que usa el algoritmo de retropropagación.

- Codificación de las reglas de desarrollo

Una regla de desarrollo en este método es una regla de generación similar a una regla de producción en un sistema basado en el conocimiento. El patrón de conectividad de una RNAE en forma de matriz, se construye desde una base y se le aplican repetitivamente diversas reglas de desarrollo a los elementos no terminales de la matriz actual hasta obtener elementos no terminales que determinarán si existe conexión o no. En otra variante del método, en vez de llegar a una solución verdadera o falsa, se llega a obtener unos valores reales que, de existir conexión, especificarán el peso de la misma.

***Evolución de las funciones de transferencia entre nodos***

A pesar de que la función de transferencia tiene un impacto significativo en el rendimiento de las RNAE y de que es una parte importante de su arquitectura no se han llevado a cabo muchos trabajos respecto a esta evolución.

La función de transferencia en los nodos de una RNAE puede ser diferente y generada de manera automática por una AG en lugar de por un experto. Por lo general un mismo grupo de nodos en una RNAE tienden a tener la misma función de transferencia con ligeros matices en algunos parámetros mientras que grupos de nodos diferentes pueden tener distintas funciones de transferencia.

La optimización de la regla de aprendizaje, es decir la regla que actualiza los pesos de las conexiones, tan sólo se ha tocado en algunos pocos estudios. Esta clase de búsqueda de una regla de aprendizaje óptima sólo puede realizarla los expertos mediante prueba y error gracias a su experiencia. El ajuste adaptativo de los parámetros del algoritmo de retropropagación, como la razón de aprendizaje y el momento, a lo largo de la evolución, se puede considerar un primer paso hacia la evolución de las reglas de aprendizaje.

### 2.3.3. Evolución de las reglas de aprendizaje

Se conoce que diferentes tareas de aprendizaje y arquitecturas necesitan algoritmos de entrenamiento diferentes, por ejemplo, los AG son adecuados para entrenar

RNAE con conexiones retroalimentadas y muchas capas ocultas, mientras que el algoritmo de retropropagación obtiene mejores resultados en RNAE con pocas capas ocultas.

Los algoritmos de aprendizaje disponen de varios parámetros que pueden ajustarse una vez seleccionado alguno. Este ajuste es muy complicado debido a que existe poco conocimiento sobre la arquitectura de la RNAE y la tarea de aprendizaje.

## *2.4. Predicción de series temporales con redes de neuronas artificiales*

Una de las principales aplicaciones de las RNAs es la predicción de series temporales. Las RNA son capaces de aprender y generalizar de la experiencia pasada. Trabajos con RNA han demostrado recientemente que son ideales para el reconocimiento y clasificación de patrones. Las RNA disponen de ciertas características que invitan a usarlas para la predicción de series temporales.

- Las RNA son métodos auto adaptativos conducidos por los datos. Se adaptan bien a problemas cuya solución requiere de conocimientos muy complicados pero de los que se disponen bastantes datos. El único problema aquí es que los datos, al proceder del mundo real, pueden estar contaminados con ruido.
- Las RNA, tras aprender de los datos presentados, pueden inferir la parte desconocida del conjunto incluso a pesar del ruido, pueden generalizar.
- Las RNA son aproximadores funcionales universales, se ha demostrado que pueden aproximar cualquier función continua a cualquier precisión.
- Las RNA son no lineales como la mayoría de los sistemas del mundo real. Son capaces de llevar a cabo un modelado no lineal sin conocer previamente las relaciones entre las variables de entrada y de salida.

## *2.5. Modelo ADANN: Diseño Automático de Redes Neuronales Artificiales*

El modelo ADANN, propuesto por J. Peralta, G. Gutierrez y A. Sanchis en [1], se trata de una aproximación inicial al diseño automático de RNAs mediante el uso de AGs para predecir series temporales.

El problema de diseñar una RNA se puede ver como un problema de búsqueda de la RNA más adecuada dentro de todo el espacio de RNAs.

A la hora de diseñar la RNA se establece que el mejor tipo de red para resolver la tarea de predicción es el perceptrón multicapa de acuerdo a [2], centrándose en el perceptrón multicapa totalmente conectado y con una única capa de oculta, y el algoritmo de aprendizaje el de retropropagación. El motivo de esta decisión es que las RNAs de una sola capa oculta son más fáciles de entrenar y se trabaja mejor con ellas.

Para diseñar la RNA se necesitan definir los valores de los parámetros de la misma (número de capas ocultas, número de nodos de entrada, número de nodos de salida, algoritmo de aprendizaje, algoritmo de inicialización de los pesos de las conexiones, ...) , parte de estos parámetros serán fijados con antelación mientras que otros serán seleccionados durante el proceso de búsqueda del AG. En el diseño de la RNA se han decantado por un esquema de codificación directa, es decir, que el cromosoma del AG contiene información sobre los parámetros de la topología, la arquitectura, aprendizaje, etc. En este caso se establecen los siguientes parámetros a codificar en el cromosoma:

- Número de nodos en la capa de entrada (p.e. Número de elementos anteriores de la serie temporal) (*i*). Este valor esta limitado mediante un parámetro del sistema proporcionado por el usuario.
- Número de nodos de la capa oculta (*h*). Este valor esta limitado mediante un parámetro del sistema proporcionado por el usuario.
- Razón de aprendizaje del algoritmo de retropropagación ($\alpha$). Es un valor entre 0 y 1.
- Semilla de inicialización de los pesos de las conexiones (*s*).

Tanto el parámetro "*i*" como el "*h*" están limitados mediante el parámetro "*A*" combinado con el número total de elementos que tenga la serie temporal a predecir, según se muestra en la ecuación 1

$$\text{max\_entradas} = A \times nts$$

$$\text{max\_ocultas} = 2 \times \text{max\_entradas}$$

*Ecuación 1: Función de máximo número de nodos de entrada y ocultos.*

Según sea el problema, el número de elementos conocidos de la serie temporal (nts), y haciendo uso de la función de auto correlación parcial, es decir, aquella que captura la memoria del proceso: número de periodos en los que una variable sigue teniendo alguna influencia en la evolución del proceso, el parámetro "*A*" es un parámetro introducido por el usuario.

En el AG que realiza la búsqueda cada individuo corresponde a una RNA. Los parámetros de cada RNA se codifican en el cromosoma se la siguiente manera:

- Número de nodos de entrada, valor "*i*". Son dos genes, es decir, dos dígitos decimales.
- Número de nodos ocultos, valor "*h*". Son dos genes.
- Razón de aprendizaje, valor "$\alpha$". Son dos genes.
- Semilla de inicialización de los pesos de las conexiones, valor "*s*". Son diez genes, esto es debido a que SNNS, usa un entero largo para la semilla.

Por lo tanto los parámetros "i", "h", "$\alpha$" y "s" se obtienen del cromosoma como puede verse en ecuación 2.

$$Cromosoma:$$
$$g_{i1}\, g_{i2}\, g_{h1}\, g_{h2}\, g_{\alpha 1}\, g_{\alpha 2}\, g_{s1}\, g_{s2}\, g_{s3}\, g_{s4}\, g_{s5}\, g_{s6}\, g_{s7}\, g_{s8}\, g_{s9}\, g_{s10}$$

$$0 \le g_{xy} \le 9, x = i, h, \alpha, y = 1..10$$
$$i = \max\_inputs \cdot ((g_{i1} \cdot 10 + g_{i2})/100)$$
$$h = \max\_hiddens \cdot ((g_{h1} \cdot 10 + g_{h2})/100)$$
$$\alpha = ((g_{\alpha 1} \cdot 10 + g_{\alpha 2}))/100$$
$$s = g_{s1}\, g_{s2}\, g_{s3}\, g_{s4}\, g_{s5}\, g_{s6}\, g_{s7}\, g_{s8}\, g_{s9}\, g_{s10}$$

*Ecuación 2: Codificación del cromosoma.*

El AG realiza la búsqueda de la forma siguiente:

1. Se genera de manera aleatoria un conjunto de cromosomas, generación inicial.
2. Por cada cromosoma se obtiene su fenotipo (p.e. las arquitecturas de las RNAs) y su valor de adecuación. Para esto:
   a) El fenotipo de un individuo de la generación actual se obtiene haciendo uso de SNNS.
   b) El número de nodos de entrada que tenga cada individuo determina sus conjuntos de test y de entrenamiento.
   c) La red se entrena con el algoritmo de retropropagación usando SNNS. El fenotipo final de cada individuo es la arquitectura (topología más pesos de las conexiones) de la red cuando el error de test es mínimo durante el entrenamiento.
3. Para generar la siguiente generación de individuos se aplican los operadores evolutivos: el elitismo, la selección, el sobre-cruzamiento y la mutación una vez que se ha obtenido el valor de adecuación.
4. Los pasos 2 y 3 se repiten hasta que se alcanza un número máximo de generaciones definidas por el usuario.

En las figuras 5 y 6 podemos ver un esquema del proceso.

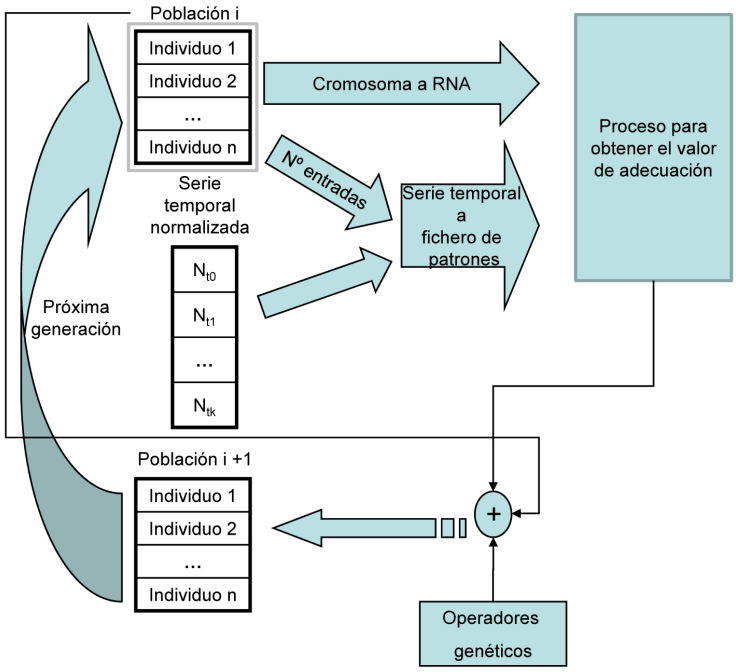


*Figura 5: ADANN.*

El error mínimo de test durante el proceso de aprendizaje nos dará la función de adecuación (*fitness*).

En el desarrollo del presente trabajo se ha experimentado, con el fin de lograr una mejora significativa en la predicción de resultados, con los siguientes parámetros del algoritmo genético:

- Se ha incrementado el número de individuos que componen la población en comparación con experimentos anteriores.
- El número de generaciones por experimento se ha aumentado de 100 a 250.

También se ha modificado el algoritmo del operador genético de selección codificando los siguientes:

- Torneo de 3 padres.

- Ruleta

Los demás parámetros se han dejado como sigue a continuación:

- Porcentaje de elitismo, aquel porcentaje de los mejores individuos que permanecen sin cambios en la siguiente generación, 10%.
- El método de sobrecruzamiento se lleva a cabo cortando los cromosomas en un punto medio aleatorio y juntando la primera parte del padre con la segunda de la madre y viceversa, con lo que obtendrán dos nuevos descendientes.
- La probabilidad de mutación es uno partido por la longitud del cromosoma (1/long_cromosoma=0,7), llevándola a cabo por cada cromosoma.

Una vez que el AG ha completado las generaciones indicadas, la mejor RNA obtenida de los individuos de la última generación se usa para predecir los valores desconocidos de la serie temporal, el conjunto de validación.

Los valores desconocidos (*t+1*) se predecirán uno a uno usando los "*k*" valores previos conocidos (*t, t-1, …, t-k*). Esto implica que para predecir varios valores consecutivos (*t+1, t+2, …*), cada vez que se predice un nuevo valor, este se tiene que incluir en la serie temporal siguiendo el orden correspondiente para así poder predecir el siguiente valor.

La razón por la que solo se usa un único nodo en la capa de salida es porque si se intentara predecir más de un valor a la vez (*t+1, t+2, …, t+n*), el instante *t+1* obtendría una predicción razonable, pero si nos fijamos en *t+2*, este valor tan solo podría usar los valores desde *1* hasta *t*, con lo que la información que pudiera aportarle *t+1* en su predicción se perdería, y así sucesivamente para *t+3* que perdería la información que le pudiera aportar *t+1* y *t+2* en su predicción.

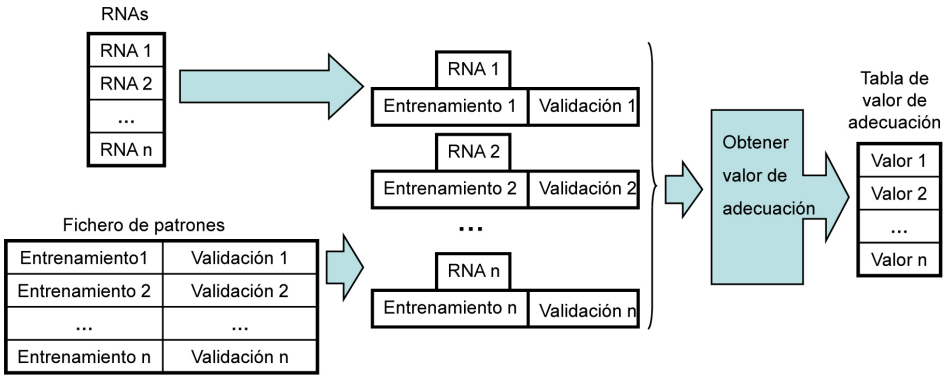


*Figura 6: ADANN.*

## *2.6. Herramientas usadas*

Para el desarrollo de este proyecto se han usado exclusivamente herramientas pertenecientes al software libre, desde el sistema operativo hasta la herramienta de planificación de proyectos. En los sucesivos puntos veremos el software libre usado más destacable.

### 2.6.1. Linux Fedora

Es un sistema operativo libre, una distribución de software libre de propósito general basada en el núcleo Linux. Linux es un núcleo de sistema operativo libre tipo Unix. Es uno de los principales ejemplos de software libre. Linux está licenciado bajo la GPL v2 y está desarrollado por colaboradores de todo el mundo.

Además del núcleo Linux, la distribuciones incluye bibliotecas y herramientas del proyecto GNU y el sistema de ventanas X Window System que facilita la interacción con el usuario.

Este sistema operativo es el que se ha usado para la ejecución del resto de herramientas utilizadas para llevar a cabo el proyecto presentado.

### 2.6.2. LibreOffice

LibreOffice es una suite gratuita de productividad personal de código abierto para

Windows, Macintosh y Linux, que ofrece 6 aplicaciones ricas en funcionalidades para todas sus necesidades de producción de documentos y procesamiento de datos: Writer (procesador de textos), Calc (hoja de cálculo), Impress (crear presentaciones), Draw (editor de diagramas y dibujos), Base (bases de datos) y Math (editor de ecuaciones). La asistencia y documentación es gratuita gracias a su gran comunidad, usuarios dedicados, colaboradores y desarrolladores.

***Writer***

Es el procesador de textos dentro de LibreOffice. Se usa para todo, desde garabatear una carta rápida a producir un libro entero con las tablas de contenido, ilustraciones embebidas, bibliografía y diagramas. Es muy potente y totalmente gratuito al igual que todas las aplicaciones que componen LibreOffice. Es compatible con Microsoft Word.

Mediante el Writer se ha escrito y maquetado la presente memoria.

***Calc***

Es la hoja de cálculo integrada en la suite LibreOffice gratuita y de código abierto. Es compatible con Microsoft Excel. Al igual que Writer, Calc es muy potente y permite hacer un montón de funciones como la generación de gráficas entre otras.

Todos los resultados obtenidos en los experimentos han sido tratados mediante el Calc. También ha servido para generar las gráficas y tablas mostradas a lo largo de la memoria.

***Impress***

Impress es una herramienta para crear presentaciones multimedia efectivas. Sus presentaciones pueden ser mejorada con imágenes prediseñadas en 2D y 3D, efectos especiales y estilos de transición, animaciones y herramientas de dibujo de alto impacto.

Impress ha sido usado para hacer la presentación de esta memoria.

### 2.6.3. Netbeans IDE

NetBeans es un entorno de desarrollo integrado libre, hecho principalmente para el lenguaje de programación Java aunque actualmente soporta otros lenguajes como C, C++ , PHP, etc. Existe además un número importante de módulos para extenderlo. NetBeans IDE es un producto libre y gratuito sin restricciones de uso.

NetBeans es un proyecto de código abierto de gran éxito con una gran base de usuarios, una comunidad en constante crecimiento, y con cerca de 100 socios en todo el mundo. Sun MicroSystems fundó el proyecto de código abierto NetBeans en junio de 2000 y continúa siendo el patrocinador principal de los proyectos.

Netbeans se ha usado inicialmente como entorno de desarrollo en C para la codificación de los algoritmos de selección de 3 padres y de la ruleta y, posteriormente, como entorno de desarrollo en Java para el desarrollo de la herramienta *GetForecastedValues*.

### 2.6.4. SNNS

SNNS (Stuttgart Neural Network Simulator) es un simulador por software de redes de neuronas en sistemas Unix desarrollado en el Instituto de Sistemas de Alto Rendimiento Paralelos y Distribuidos (IPVR) de la Universidad de Stuttgart. El objetivo del proyecto SNNS es crear un entorno de simulación flexible y eficaz para investigación y aplicación de RNAs.

El simulador SNNS esta compuestos por dos componentes principales:

- el núcleo del simulador escrito en C
- la interfaz gráfica de usuario bajo X11R4 o X11R5

El núcleo del simulador opera en las estructuras de datos internas de las RNAs y realiza todas las operaciones para aprender y recordar. También se puede usar en programas de usuario sin las otras partes como un programa C embebido. Es compatible con cualquier tipo de topología y, como RCS (Ro*chester* Connectionist Simulator), soporta

el concepto de sitios. SNNS puede ser ampliado con funciones de activación, de salida, de sitio y procedimientos de aprendizaje definidos por el usuario, que son simples programas en C enlazados al núcleo del simulador.

Aunque al final se ha usado SNNS para el desarrollo del modelo ADANN por su facilidad a la hora de generar, entrenar y verificar las redes neuronales mediante sencillos scripts, se podría haber tratado de usar otros como Aspirin/MIGRAINE, GENESIS, PlaNet, RCS, Xerion, YANNS,...

A través de SNNS, en concreto de sus programas batchman y ff_bignet, se han entrenado y probado las RNA generadas por los AG y obtenido los resultados finales.

### 2.6.5. Java SUN

Java es un lenguaje de programación orientado a objetos, desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria. Con respecto a la memoria, su gestión no es un problema ya que ésta es gestionada por el propio lenguaje y no por el programador.

Las aplicaciones Java están típicamente compiladas en un *bytecode*, aunque la compilación en código máquina nativo también es posible. En el tiempo de ejecución, el *bytecode* es normalmente interpretado o compilado a código nativo para la ejecución, aunque la ejecución directa por hardware del *bytecode* por un procesador Java también es posible.

La implementación original y de referencia del compilador, la máquina virtual y las bibliotecas de clases de Java fueron desarrollados por Sun Microsystems en 1995. Desde entonces, Sun ha controlado las especificaciones, el desarrollo y evolución del lenguaje a través del Java Community Process, si bien otros han desarrollado también implementaciones alternativas de estas tecnologías de Sun, algunas incluso bajo licencias de software libre.

Entre diciembre de 2006 y mayo de 2007, Sun Microsystems liberó la mayor parte de sus tecnologías Java bajo la licencia GNU GPL, de acuerdo con las especificaciones del Java Community Process, de tal forma que prácticamente todo el Java de Sun es ahora software libre.

Este lenguaje se ha usado para la implementación de la herramienta *GetForecastedValues*.

### 2.6.6. GCC (GNU Compiler Collection)

La colección de compiladores de GNU incluyen interfaces para C, C++, Objetive-C, Fortran, Java, Ada así como librerías para estos lenguajes. GCC inicialmente fue escrito como el compilador para el sistema operativo GNU. El sistema GNU fue desarrollado para ser totalmente software libre, en el sentido de que respete la libertad del usuario.

Este compilador es el usado para la generación en ADANN del AG, por lo que se ha tenido que usar para añadir los nuevos algoritmos de selección torneo de 3 padres y ruleta.

### 2.6.7. Planner

Planner es la herramienta de gestión de proyectos de GNOME. Su meta es ser una aplicación de gestión de proyectos multi-plataforma y sencilla de usar.

Planner se ha desarrollado en C como una aplicación GTK+ y bajo licencia GPLv2 o cualquier versión posterior. Puede almacenar sus datos tanto en XML como en una base de datos Postgress. Los proyectos gestionados se pueden imprimir en PDF o exportados a HTML para visualizarlos correctamente en cualquier navegador.

Por último, mediante Planner se ha llevado la gestión del proyecto y se han obtenido los diagramas de GANNT que se han visto en el apartado 4.2.

# 3 GESTIÓN DEL PROYECTO

En este apartado se va a describir la manera en la que se ha planificado el trabajo desarrollado en el proyecto.

## *3.1. Definición de tareas*

Para llevar a cabo el proyecto se han enumerado la lista de tareas a realizar durante el ciclo de vida del mismo, los recursos asignados a cada tarea y el tiempo que le llevará al recurso completar la tarea asignada.

El listado de recursos, tanto materiales como humanos, disponibles para la realización del trabajo se enumera a continuación:

- MALOBATO: la persona que ha desarrollado el proyecto y que debido a su situación laboral y familiar tan sólo puede dedicar 3 horas al día. Su jornada laboral abarca desde las 16:30 hasta las 19:30 de la tarde, por lo que la duración de sus tareas se reflejará en el diagrama de Gannt de manera acorde a este horario.
- CPU1: Un núcleo de la CPU de doble núcleo sobre la que se lanzan los experimentos.
- CPU2: El otro núcleo de la máquina de doble núcleo sobre la que se lanzan los experimentos.

Los dos últimos recursos, CPU1 y CPU2, son recursos materiales por lo que no se cansan y su jornada de trabaja no tiene límites. En el proyecto realizan sus tareas durante las 24 horas del día sin descanso.

Seguidamente se detallan las fases junto con las tareas definidas en la planificación del proyecto:

*1* Aprendizaje del sistema

*1.1* Documentación: Recopilación y estudio de documentación relativa a los temas y técnicas abordadas en el proyecto inteligencia artificial, RNA, AG, etc, así como de la herramienta SNNS para conocer como se integra con la aplicación, cómo definir guiones de trabajo y como ejecutarlos a través de "*scripts*" y bajo línea de comandos.

*1.2* Conceptos de funcionamiento: estudio del Modelo ADANN para conocer como se ha planteado, cómo funciona, cómo se ejecuta y cómo se extraen sus resultados.

*1.3* Código fuente: Estudio del código fuente de la parte del algoritmo genético implementada en el modelo ADANN para poder ampliar el modelo con los nuevos algoritmos de selección planteados.

*2* Documentación del proyecto: Tarea abierta durante todo el desarrollo del proceso donde se documenta la memoria de todo lo que se va realizando durante el proyecto.

*3* Desarrollo de la herramienta "*GetForecastedValues*": diseño y desarrollo de la utilidad de lanzamiento y extracción automática de resultados de los experimentos generados con el modelo ADANN.

*3.1* Análisis: Planteamiento de los requerimientos que tendrá la aplicación.

*3.2* Diseño: Especificación de como se desarrollarán cada uno de los requisitos.

*3.3* Programación: Desarrollo de la utilidad requisito a requisito.

*3.4* Pruebas: Comprobación de que la aplicación desarrollada satisface cada uno de los requisitos planteados.

*4* Experimento torneo de 2 padres con 20 individuos y 100 generaciones repetido 10 veces. Lanzamiento de los experimentos usando el algoritmo de selección torneo, con una población de 20 individuos y 100 generaciones. Este experimento se repite 10 veces por cada una de las cinco series temporales sobre las que se realiza la

experimentación.

*4.1* Experimento con la serie Dow-Jones. Lanzamiento del experimento con la serie temporal Dow-Jones.

*4.2* Experimento con la serie Passengers. Lanzamiento del experimento con la serie temporal Passengers.

*4.3* Experimento con la serie Temperature. Lanzamiento del experimento con la serie temporal Temperature.

*4.4* Experimento con la serie Mackey-Glass. Lanzamiento del experimento con la serie temporal Mackey-Glass.

*4.5* Experimento con la serie Quebec. Lanzamiento del experimento con la serie temporal Quebec.

*5* Extracción de los resultados obtenidos en el experimento del punto 4. En esta tarea se prueban las RNAs obtenidas en los experimentos y se extraen los resultados obtenidos.

*6* Experimento torneo de 2 padres con 50 individuos y 100 generaciones repetido 10 veces. Lanzamiento de los experimentos usando el algoritmo de selección torneo, con una población de 50 individuos y 100 generaciones. Este experimento se repite 10 veces por cada una de las cinco series temporales sobre las que se realiza la experimentación.

*6.1* Experimento con la serie Dow-Jones. Lanzamiento del experimento con la serie temporal Dow-Jones.

*6.2* Experimento con la serie Passengers. Lanzamiento del experimento con la serie temporal Passengers.

*6.3* Experimento con la serie Temperature. Lanzamiento del experimento con la serie temporal Temperature.

*6.4* Experimento con la serie Mackey-Glass. Lanzamiento del experimento con

la serie temporal Mackey-Glass.

*6.5* Experimento con la serie Quebec. Lanzamiento del experimento con la serie temporal Quebec.

*7* Extracción de los resultados obtenidos en el experimento del punto 6. En esta tarea se prueban las RNAs obtenidas en los experimentos y se extraen los resultados obtenidos.

*8* Codificación del algoritmo de selección del AG torneo de 3 padres.

*9* Experimento torneo de 2 padres con 20 individuos y 250 generaciones repetido 10 veces. Lanzamiento de los experimentos usando el algoritmo de selección torneo, con una población de 20 individuos y 250 generaciones. Este experimento se repite 10 veces por cada una de las cinco series temporales sobre las que se realiza la experimentación.

*9.1* Experimento con la serie Dow-Jones. Lanzamiento del experimento con la serie temporal Dow-Jones.

*9.2* Experimento con la serie Passengers. Lanzamiento del experimento con la serie temporal Passengers.

*9.3* Experimento con la serie Temperature. Lanzamiento del experimento con la serie temporal Temperature.

*9.4* Experimento con la serie Mackey-Glass. Lanzamiento del experimento con la serie temporal Mackey-Glass.

*9.5* Experimento con la serie Quebec. Lanzamiento del experimento con la serie temporal Quebec.

*10* Extracción de los resultados obtenidos en el experimento del punto 9. En esta tarea se prueban las RNAs obtenidas en los experimentos y se extraen los resultados obtenidos.

*11* Codificación del algoritmo de selección del AG de la ruleta.

*12* Experimento torneo de 3 padres con 20 individuos y 100 generaciones repetido 10 veces. Lanzamiento de los experimentos usando el algoritmo de selección torneo, de 3 padres con una población de 20 individuos y 100 generaciones. Este experimento se repite 10 veces por cada una de las cinco series temporales sobre las que se realiza la experimentación.

*12.1* Experimento con la serie Dow-Jones. Lanzamiento del experimento con la serie temporal Dow-Jones.

*12.2* Experimento con la serie Passengers. Lanzamiento del experimento con la serie temporal Passengers.

*12.3* Experimento con la serie Temperature. Lanzamiento del experimento con la serie temporal Temperature.

*12.4* Experimento con la serie Mackey-Glass. Lanzamiento del experimento con la serie temporal Mackey-Glass.

*12.5* Experimento con la serie Quebec. Lanzamiento del experimento con la serie temporal Quebec.

*13* Extracción de los resultados obtenidos en el experimento del punto 12. En esta tarea se prueban las RNAs obtenidas en los experimentos y se extraen los resultados obtenidos.

*14* Experimento ruleta con 20 individuos y 100 generaciones repetido 10 veces. Lanzamiento de los experimentos usando el algoritmo de selección de la ruleta con una población de 20 individuos y 100 generaciones. Este experimento se repite 10 veces por cada una de las cinco series temporales sobre las que se realiza la experimentación.

*14.1* Experimento con la serie Dow-Jones. Lanzamiento del experimento con la serie temporal Dow-Jones.

*14.2* Experimento con la serie Passengers. Lanzamiento del experimento con la serie temporal Passengers.

*14.3* Experimento con la serie Temperature. Lanzamiento del experimento con la serie temporal Temperature.

*14.4* Experimento con la serie Mackey-Glass. Lanzamiento del experimento con la serie temporal Mackey-Glass.

*14.5* Experimento con la serie Quebec. Lanzamiento del experimento con la serie temporal Quebec.

*15* Extracción de los resultados obtenidos en el experimento del punto 14. En esta tarea se prueban las RNAs obtenidas en los experimentos y se extraen los resultados obtenidos.

*16* Conclusiones generales obtenidas comparando los resultados generados por cada experimentación con el resto. Preparación de hojas de datos para analizar los resultados obtenidos y poder comparar todos los experimentos lanzados.

*17* Presentación: se genera la presentación que servirá de guía para la presentación del proyecto.

## *3.2. Diagramas de Gannt*

Con todas estas actividades se desarrollo una planificación inicial que se puede ver reflejada en la figura 7. Así mismo, en la figura 8 se pueden ver las diferentes tareas estimadas con sus duraciones y recursos asignados.

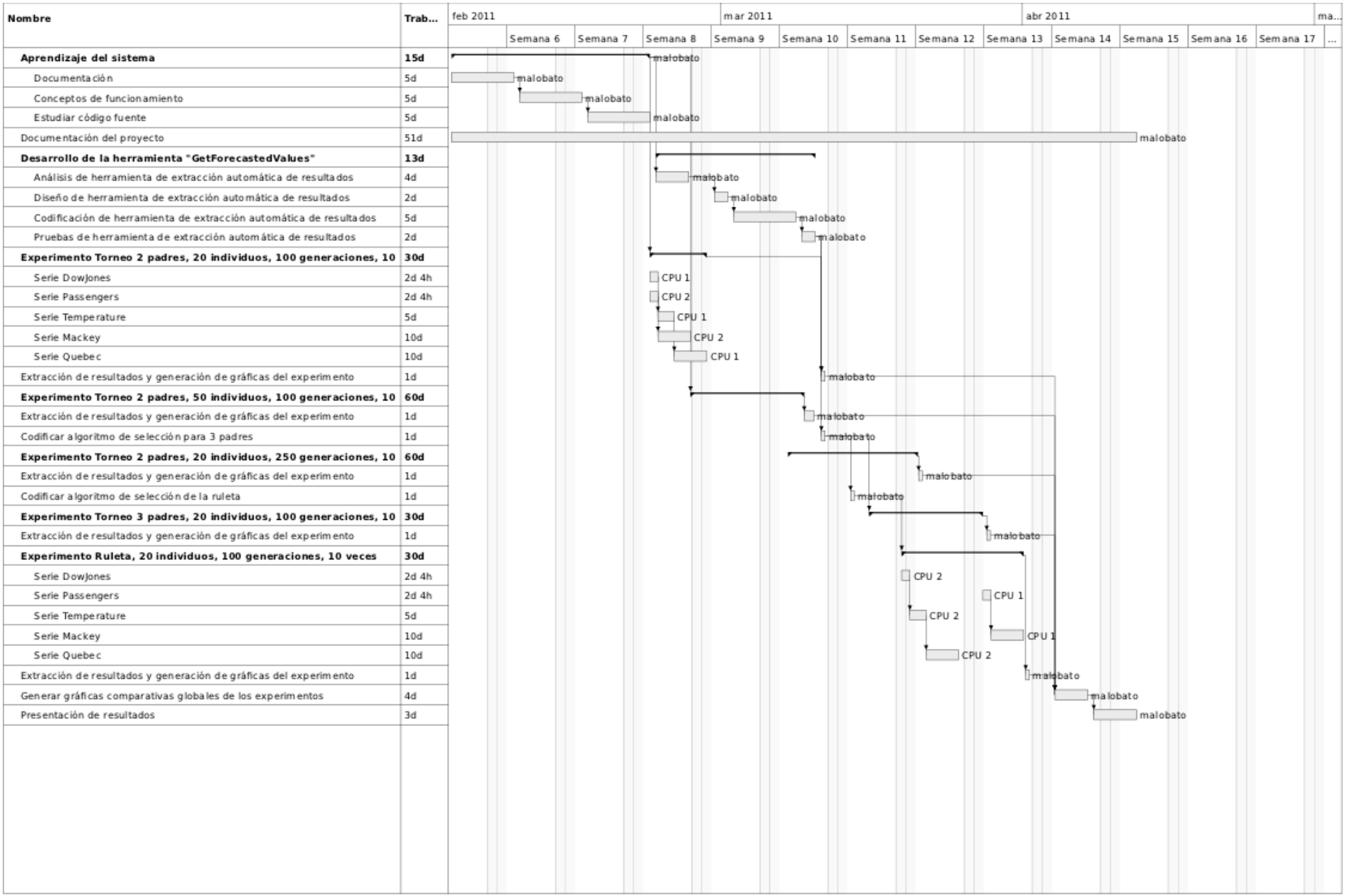


*Figura 7: Diagrama de Gannt inicial del proyecto.*

| WBS | Nombre | Inicio | Fin | Asignado a |
|---|---|---|---|---|
| 1 | **Aprendizaje del sistema** | **feb 1** | **feb 21** | malobato |
| 1.1 | Documentación | feb 1 | feb 7 | malobato |
| 1.2 | Conceptos de funcionamiento | feb 8 | feb 14 | malobato |
| 1.3 | Estudiar código fuente | feb 15 | feb 21 | malobato |
| 2 | Documentación del proyecto | feb 1 | abr 12 | malobato |
| 3 | **Desarrollo de la herramienta "GetForecastedValues"** | **feb 22** | **mar 10** | |
| 3.1 | Análisis de herramienta de extracción automática de resultados | feb 22 | feb 25 | malobato |
| 3.2 | Diseño de herramienta de extracción automática de resultados | feb 28 | mar 1 | malobato |
| 3.3 | Codificación de herramienta de extracción automática de resultados | mar 2 | mar 8 | malobato |
| 3.4 | Pruebas de herramienta de extracción automática de resultados | mar 9 | mar 10 | malobato |
| 4 | **Experimento Torneo 2 padres, 20 individuos, 100 generaciones, 10 veces** | **feb 21** | **feb 27** | |
| 4.1 | Serie DowJones | feb 21 | feb 22 | CPU 1 |
| 4.2 | Serie Passengers | feb 21 | feb 22 | CPU 2 |
| 4.3 | Serie Temperature | feb 22 | feb 24 | CPU 1 |
| 4.4 | Serie Mackey | feb 22 | feb 25 | CPU 2 |
| 4.5 | Serie Quebec | feb 24 | feb 27 | CPU 1 |
| 5 | Extracción de resultados y generación de gráficas del experimento | mar 11 | mar 11 | malobato |
| 6 | **Experimento Torneo 2 padres, 50 individuos, 100 generaciones, 10 veces** | **feb 25** | **mar 9** | |
| 6.1 | Serie DowJones | feb 27 | mar 1 | CPU 1 |
| 6.2 | Serie Passengers | feb 25 | feb 27 | CPU 2 |
| 6.3 | Serie Temperature | feb 27 | mar 2 | CPU 2 |
| 6.4 | Serie Mackey | mar 1 | mar 7 | CPU 1 |
| 6.5 | Serie Quebec | mar 2 | mar 9 | CPU 2 |
| 7 | Extracción de resultados y generación de gráficas del experimento | mar 9 | mar 10 | malobato |
| 8 | Codificar algoritmo de selección para 3 padres | mar 11 | mar 11 | malobato |
| 9 | **Experimento Torneo 2 padres, 20 individuos, 250 generaciones, 10 veces** | **mar 7** | **mar 21** | |
| 9.1 | Serie DowJones | mar 7 | mar 9 | CPU 1 |
| 9.2 | Serie Passengers | mar 9 | mar 11 | CPU 1 |
| 9.3 | Serie Temperature | mar 11 | mar 14 | CPU 1 |
| 9.4 | Serie Mackey | mar 14 | mar 21 | CPU 1 |
| 9.5 | Serie Quebec | mar 9 | mar 16 | CPU 2 |
| 10 | Extracción de resultados y generación de gráficas del experimento | mar 21 | mar 21 | malobato |
| 11 | Codificar algoritmo de selección de la ruleta | mar 14 | mar 14 | malobato |
| 12 | **Experimento Torneo 3 padres, 20 individuos, 100 generaciones, 10 veces** | **mar 16** | **mar 27** | |
| 12.1 | Serie DowJones | mar 21 | mar 22 | CPU 1 |
| 12.2 | Serie Passengers | mar 22 | mar 22 | CPU 1 |
| 12.3 | Serie Temperature | mar 22 | mar 24 | CPU 1 |
| 12.4 | Serie Mackey | mar 24 | mar 27 | CPU 1 |
| 12.5 | Serie Quebec | mar 16 | mar 19 | CPU 2 |
| 13 | Extracción de resultados y generación de gráficas del experimento | mar 28 | mar 28 | malobato |
| 14 | **Experimento Ruleta, 20 individuos, 100 generaciones, 10 veces** | **mar 19** | **abr 1** | |
| 14.1 | Serie DowJones | mar 19 | mar 20 | CPU 2 |
| 14.2 | Serie Passengers | mar 27 | mar 28 | CPU 1 |
| 14.3 | Serie Temperature | mar 20 | mar 22 | CPU 2 |
| 14.4 | Serie Mackey | mar 28 | abr 1 | CPU 1 |
| 14.5 | Serie Quebec | mar 22 | mar 25 | CPU 2 |
| 15 | Extracción de resultados y generación de gráficas del experimento | abr 1 | abr 1 | malobato |
| 16 | Generar gráficas comparativas globales de los experimentos | abr 4 | abr 7 | malobato |
| 17 | Presentación de resultados | abr 8 | abr 12 | malobato |

*Figura 8: Lista de tareas planificadas inicialmente.*

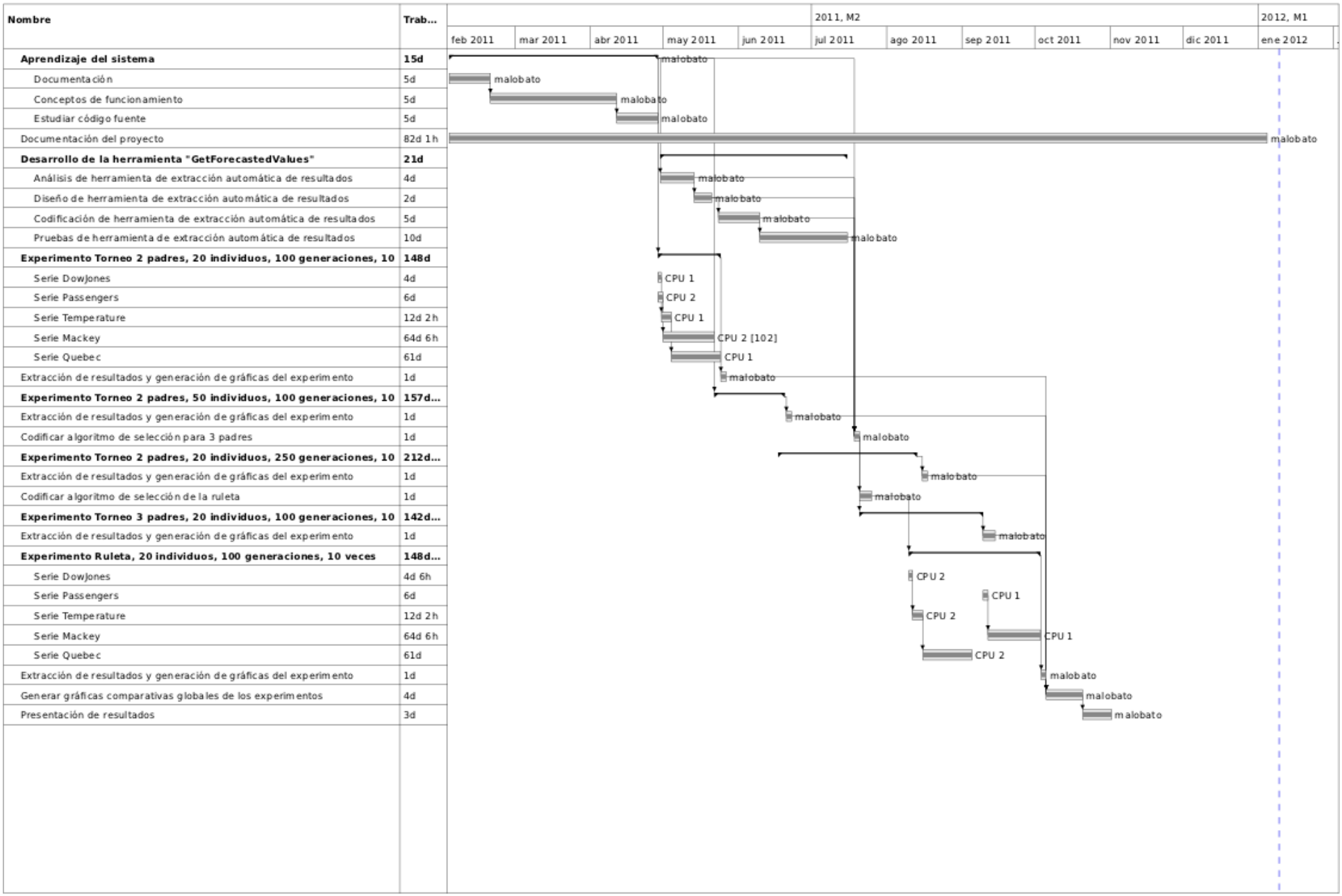


*Figura 9: Diagrama de Gannt final del proyecto.*

| WBS | Nombre | Inicio | Fin | Asignado a |
|---|---|---|---|---|
| 1 | **Aprendizaje del sistema** | **feb 1** | **abr 28** | malobato |
| 1.1 | Documentación | feb 1 | feb 18 | malobato |
| 1.2 | Conceptos de funcionamiento | feb 18 | abr 11 | malobato |
| 1.3 | Estudiar código fuente | abr 11 | abr 28 | malobato |
| 2 | Documentación del proyecto | feb 1 | ene 4 | malobato |
| 3 | **Desarrollo de la herramienta "GetForecastedValues"** | **abr 29** | **jul 15** | |
| 3.1 | Análisis de herramienta de extracción automática de resultados | abr 29 | may 13 | malobato |
| 3.2 | Diseño de herramienta de extracción automática de resultados | may 13 | may 20 | malobato |
| 3.3 | Codificación de herramienta de extracción automática de resultados | may 23 | jun 9 | malobato |
| 3.4 | Pruebas de herramienta de extracción automática de resultados | jun 9 | jul 15 | malobato |
| 4 | **Experimento Torneo 2 padres, 20 individuos, 100 generaciones, 10 veces** | **abr 28** | **may 24** | |
| 4.1 | Serie DowJones | abr 28 | abr 30 | CPU 1 |
| 4.2 | Serie Passengers | abr 28 | abr 30 | CPU 2 |
| 4.3 | Serie Temperature | abr 30 | may 4 | CPU 1 |
| 4.4 | Serie Mackey | abr 30 | may 22 | CPU 2 |
| 4.5 | Serie Quebec | may 4 | may 24 | CPU 1 |
| 5 | Extracción de resultados y generación de gráficas del experimento | may 24 | may 26 | malobato |
| 6 | **Experimento Torneo 2 padres, 50 individuos, 100 generaciones, 10 veces** | **may 22** | **jun 19** | |
| 6.1 | Serie DowJones | may 24 | may 27 | CPU 1 |
| 6.2 | Serie Passengers | may 22 | may 25 | CPU 2 |
| 6.3 | Serie Temperature | may 25 | jun 9 | CPU 2 |
| 6.4 | Serie Mackey | may 27 | jun 17 | CPU 1 |
| 6.5 | Serie Quebec | jun 9 | jun 19 | CPU 2 |
| 7 | Extracción de resultados y generación de gráficas del experimento | jun 20 | jun 22 | malobato |
| 8 | Codificar algoritmo de selección para 3 padres | jul 18 | jul 20 | malobato |
| 9 | **Experimento Torneo 2 padres, 20 individuos, 250 generaciones, 10 veces** | **jun 17** | **ago 13** | |
| 9.1 | Serie DowJones | jun 17 | jun 21 | CPU 1 |
| 9.2 | Serie Passengers | jun 21 | jun 25 | CPU 1 |
| 9.3 | Serie Temperature | jun 25 | jul 20 | CPU 1 |
| 9.4 | Serie Mackey | jul 20 | ago 13 | CPU 1 |
| 9.5 | Serie Quebec | jun 19 | jul 3 | CPU 2 |
| 10 | Extracción de resultados y generación de gráficas del experimento | ago 15 | ago 17 | malobato |
| 11 | Codificar algoritmo de selección de la ruleta | jul 20 | jul 25 | malobato |
| 12 | **Experimento Torneo 3 padres, 20 individuos, 100 generaciones, 10 veces** | **jul 20** | **sep 9** | |
| 12.1 | Serie DowJones | ago 13 | ago 15 | CPU 1 |
| 12.2 | Serie Passengers | ago 15 | ago 17 | CPU 1 |
| 12.3 | Serie Temperature | ago 17 | ago 20 | CPU 1 |
| 12.4 | Serie Mackey | ago 20 | sep 9 | CPU 1 |
| 12.5 | Serie Quebec | jul 20 | ago 10 | CPU 2 |
| 13 | Extracción de resultados y generación de gráficas del experimento | sep 9 | sep 14 | malobato |
| 14 | **Experimento Ruleta, 20 individuos, 100 generaciones, 10 veces** | **ago 10** | **oct 3** | |
| 14.1 | Serie DowJones | ago 10 | ago 11 | CPU 2 |
| 14.2 | Serie Passengers | sep 9 | sep 11 | CPU 1 |
| 14.3 | Serie Temperature | ago 11 | ago 15 | CPU 2 |
| 14.4 | Serie Mackey | sep 11 | oct 3 | CPU 1 |
| 14.5 | Serie Quebec | ago 15 | sep 5 | CPU 2 |
| 15 | Extracción de resultados y generación de gráficas del experimento | oct 3 | oct 5 | malobato |
| 16 | Generar gráficas comparativas globales de los experimentos | oct 5 | oct 20 | malobato |
| 17 | Presentación de resultados | oct 20 | nov 1 | malobato |

*Figura 10: Lista de tareas finales del proyecto.*

Tras finalizar el proyecto, en la figura 9, se muestra el diagrama de Gannt generado durante el transcurso real del proyecto. Toda esta información de puede ver resumida por tareas en la figura 10 donde aparece cada tarea con el recurso asignado y el tiempo real empleado para la realización de la tarea.

Tanto la figura 7 como la figura 9 muestran las tareas principales del proyecto y las subtareas que las comprenden, a excepción de las tareas de experimentación, de las que sólo se han desglosado la experimentación del torneo de 2 padres con 20 individuos y 100 generaciones y la experimentación de la ruleta con 20 individuos y 100 generaciones siendo el resto de subtareas de experimentación semejantes a estas pero no mostradas para que entre el diagrama completo en una única página.

Si se comparan los dos diagramas de Gannt, se pueden ver varios puntos de retraso que se comentan a continuación:

- En la planificación real ha habido un parón total en el mes de marzo debido al nacimiento de la hija de MALOBATO. Durante este tiempo todos los recursos estuvieron parados.
- Aunque las tareas iniciales se llevaron a cabo en su tiempo estimado, en la realización de las pruebas tras la codificación de la herramienta "GetForecastedValues" se sufrió un importante retraso, casi 2 semanas, debido a que las pruebas realizadas sobre la herramienta no ofrecían los resultados esperados por lo que hubo que hacer correcciones y nuevas pruebas hasta conseguir el resultado esperado.
- La estimación inicial del tiempo que iba a tomar la realización de los experimentos difiere de la final debido, principalmente, al hecho de hacer una estimación sin tener experiencia previa en este campo y por lo tanto no saber afinar correctamente el tiempo que iba a tomar la experimentación. Se puede destacar que el tiempo de computación que requiere el lanzamiento de cada experimento ha sido muy elevado, con un número mayor de recursos materiales (CPUs) el tiempo se podría haber reducido notablemente.

- Por último comentar que la tarea que abarca la documentación completa del proyecto tuvo un retraso final debido a las correcciones que se han tenido que realizar sobre la misma memoria y bajo la supervisión del profesor D. Juan Peralta.

# 4 IMPLEMENTACIÓN

Los AGs utilizan un mecanismo de selección para elegir individuos de la población e insertarlos en un subconjunto de individuos. Los individuos de este subconjunto suelen usarse para generar nuevos hijos, los cuales formarán las bases de la próxima generación. Como estos individuos del subconjunto son aquellos cuyos genes serán heredados por la próxima generación, se desea que el subconjunto este formado por individuos "de calidad". Un mecanismo de selección en AGs es simplemente un proceso que favorece la selección del subconjunto de los mejores individuos de la población. La presión de selección es el grado según el cual los mejores individuos son favorecidos: cuanto más alta sea la presión de selección más se favorecen los mejores individuos. Esta presión de selección conduce al AG a mejorar la función de adecuación de la población en generaciones futuras. El nivel de convergencia de un AG esta determinado en gran parte por la presión de selección, presiones de selección altas llevan a niveles de convergencia altos. Los AGs son capaces de identificar soluciones óptimas o muy óptimas bajo un amplio rango de presiones de selección. Sin embargo, si la presión de selección es demasiado baja, el nivel de convergencia será muy lento, y el AG se tomará más tiempo para encontrar una solución óptima. Si la presión de selección es demasiado alta, hay bastantes posibilidades de que el AG converja a una solución incorrecta.

## *4.1. Modificaciones sobre ADANN*

La versión utilizada durante el desarrollo de este proyecto de la implementación del AG usado en el modelo ADANN, usa como método de selección el torneo de dos padres. Este método selecciona aleatoriamente a dos parejas de individuos de la población, de los cuales se queda con los individuos con la mejor adecuación de cada pareja. Estos individuos seleccionados son los que se reproducirán para la nueva generación.

El algoritmo de selección del torneo da una presión de selección manteniendo un

torneo entre n competidores, siendo n el tamaño del torneo. El ganador del torneo es el individuo con la mayor adecuación de los n competidores del torneo y se introduce en el subconjunto de los mejores individuos. El subconjunto, compuesto por los ganadores del torneo, tiene una mayor adecuación de media que la media de la adecuación de la población. Esta diferencia de adecuación da la presión de selección, que conduce al AG a mejorar la adecuación con cada generación futura. El incremento en la presión de selección se puede dar simplemente incrementando el tamaño del torneo n, ya que el ganador de un gran torneo tendrá, de media, una mayor adecuación que el ganador de un torneo pequeño.

Dado que el modelo ADANN ha sido codificado en C, los algoritmos han de ser codificados en el mismo lenguaje ampliando el módulo del AG: "*genetic_algorithm.c*" y "*genetic_algorithm.h*".

Como propuesta de posible mejora del modelo ADANN se ha optado por la codificación de dos algoritmos alternativos como método de selección del AG: el torneo de tres padres y la ruleta.

### 4.1.1. Torneo de tres padres

La implementación de este algoritmo se basa en el torneo de dos padres pero tomando tres parejas de padres diferentes. Esto permite ampliar el campo de búsqueda y seleccionar a los dos mejores individuos de las tres parejas, aquellos cuya adecuación sea mayor. Con este algoritmo se aumenta la presión de selección en una unidad ya que pasamos de un valor n=2 a un valor n=3.

La explicación del algoritmo es la siguiente:

- Seleccionar tres parejas de individuos diferentes de toda la población.
- Coger, por cada pareja, el individuo con la adecuación más alta.
- Coger de los tres individuos restantes los dos con mejor adecuación y devolverlos para el siguiente paso del AG, la reproducción.

En el listado 1 se puede ver el código fuente del algoritmo implementado en lenguage C.

```
1.  void selection3( float vector_fitness[MAX_NUMBER_INDIVIDUALS], int *padre_1, int *padre_2, int population_size )
2.  {
3.    int firstBestFather = 0; // Primer padre con el mejor fitness
4.    int secondBestFather = 0; // Segundo padre con el mejor fitness
5.    int fathers[ 6 ]; // 6 padres aleatorios y distintos
6.    int i = 0;
7.
8.    // Obtiene 6 padres aleatorios y distintos
9.    getNFathers( 6, fathers, population_size );
10.
11.   // Recorre todos los padres para ver cual tiene una mejor fitness
12.   for ( i = 1; i < 6; i++ )
13.   {
14.     if ( vector_fitness[ fathers[ i ] ] < vector_fitness[ fathers[ firstBestFather ] ] )
15.       firstBestFather = i;
16.   }
17.
18.   // Si el primer mejor padre es el primero coge el siguiente sino nadie le ganará
19.   if ( firstBestFather == 0 ) secondBestFather = 1;
20.
21.   for ( i = 1; i < 6; i++ )
22.   {
23.     // Coge el segundo mejor no el primero que ya tenemos
24.     if ( i != firstBestFather )
25.     {
26.       if ( vector_fitness[ fathers[ i ] ] < vector_fitness[ fathers[ secondBestFather ] ] )
27.         secondBestFather = i;
28.     }
29.   }
30.
31.   *padre_1 = fathers[ firstBestFather ];
32.   *padre_2 = fathers[ secondBestFather ];
33. }
```

***Listado 1: Implementación del torneo de 3 padres.***

La función *selection3* cuyo código se muestra en el listado 1 implementa el

algoritmo de selección del torneo de 3 padres. A continuación se explica el desarrollo.

La función espera cuatro parámetros:

- *vector_fitness. El* vector con la fitness de la población del algoritmo genético.
- **padre_1*. Variable pasada por referencia para devolver el primer padre seleccionado por el algoritmo.
- **padre_2*. Variable pasada por referencia para devolver el segundo padre seleccionado por el algoritmo.
- *population_size*. Tamaño total de la población usada por el algoritmo genético.

De las líneas 3 a 6 se declaran las variables usadas por la función.

La función *getNFathers* de la línea 9 se encarga de devolver en su parámetro *fathers* una cantidad de padres aleatorios y distintos indicados en el parámetro 1 de la función de un máximo indicado en el parámetro 3, en este caso, el número total de individuos de la población, *population_size*.

El bucle de las líneas 12 a 16, seleccionará de entre todos los padres devueltos por la función anterior el que mejor fitness posea para devolverlo como el primer padre.

De las líneas 19 a 29, seleccionará el segundo mejor padre posible teniendo en cuenta el primer padre ya seleccionado para no seleccionarlo de nuevo.

En las líneas 31 y 32 se devuelven los dos padres seleccionados.

### 4.1.2. Ruleta

Propuesto por DeJong, es posiblemente el método de selección más utilizado desde los orígenes de los AGs.

A cada uno de los individuos de la población se le asigna una parte proporcional a su adecuación en una ruleta, de tal forma que la suma de todos los porcentajes sea la unidad. En la figura 11 se puede ver una representación gráfica del algoritmo. Los mejores individuos recibirán una porción de la ruleta mayor que la recibida por los peores. Generalmente la población esta ordenada en base al ajuste por lo que las porciones más grandes se encuentran al inicio de la ruleta. Para seleccionar un individuo basta con generar un número aleatorio del intervalo [0..1] y devolver el individuo situado en esa posición de la ruleta. Esta posición se suele obtener recorriendo los individuos de la población y acumulando sus proporciones de ruleta hasta que la suma exceda el valor obtenido.

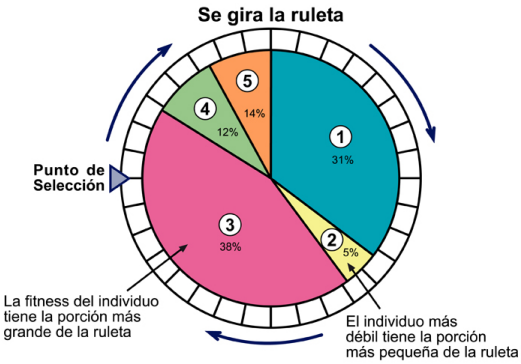


*Figura 11: Algoritmo de selección de la ruleta.*

Es un método muy sencillo, pero ineficiente a medida que aumenta el tamaño de la población, su complejidad es O( $n^2$ ). Presenta además el inconveniente de que el peor individuo puede ser seleccionado más de una vez.

En mucha bibliografía se suele denominar a este método con el nombre de la

Selección de Montecarlo.

El algoritmo se ha codificado en lenguaje C y se puede ver su implementación en el listado 2.

El código fuente de la función del listado 2 se explica a continuación.

La función roulette que implemente el algoritmo de selección de la ruleta, al igual que la función anterior espera cuatro parámetros:

- *vector_fitness. El* vector con la fitness de la población del algoritmo genético.
- **padre_1*. Variable pasada por referencia para devolver el primer padre seleccionado por el algoritmo.
- **padre_2*. Variable pasada por referencia para devolver el segundo padre seleccionado por el algoritmo.
- *population_size*. Tamaño total de la población usada por el algoritmo genético.

De la línea 3 a la línea 6 se declaran las variables usadas por la función.

De las línea 9 a la línea 16 se hace una normalización de los valores de la fitness, de tal manera que a menores valores impliquen una mejor fitness, y se calcula el sumatorio total de las fitness para luego efectuar la normalización.

En el bucle de las líneas 19 a 21 se hace la normalización de los valores de la fitness.

De las líneas 23 a 25 se calcula y almacena en la variable *fitnessS* la fitness acumulada, almacenando en el valor actual *i*, la fitness *i* la fitness acumulada anterior *i-1*. Cada uno de estos valores formarán los "quesitos" imaginarios de la ruleta que se va a hacer girar.

En la línea 28 se selecciona un valor aleatorio entre 0 y 1 que será el valor seleccionado al hacer girar la ruleta imaginaria.

```
1.  void roulette( float vector_fitness[ MAX_NUMBER_INDIVIDUALS ], int *padre_1, int *padre_2, int population_size )
2.  {
3.    float fitnessN[ population_size ]; // Fitness normalizada entre 0 y 1
4.    float fitnessS[ population_size ]; // Fitness incremental para ruleta
5.    float ft = 0.0f; // Suma total de la fitness de todos los individuos
6.    int i = 0;
7.
8.    // Fitness total para normalizar invertida porque menor fitness es mejor
9.    for ( i = 0; i < population_size; i++ )
10.   {
11.     if ( vector_fitness[ i ] > 1.0f )
12.       fitnessN[ i ] = 1.0f;
13.     else
14.       fitnessN[ i ] = vector_fitness[ i ];
15.     ft += ( 1.0f - fitnessN[ i ] );
16.   }
17.
18.   // Fitness de cada individuo normalizada
19.   for ( i = 0; i < population_size; i++ ) {
20.     fitnessN[ i ] = ( 1.0f - fitnessN[ i ] ) / ft;
21.   }
22.   // Fitness acumulada para la ruleta
23.   fitnessS[ 0 ] = fitnessN[ 0 ];
24.   for ( i = 1; i < population_size; i++ )
25.     fitnessS[ i ] = fitnessS[ i - 1 ] + fitnessN[ i ];
26.   // Selección de dos números aleatorios diferentes
27.   // Valor entre 0 y 1
28.   float selected = (float)rand() / RAND_MAX;
29.   int found = 0;
30.   for ( i = 0; i < population_size && found == 0; i++ )
31.   {
32.     if ( selected < fitnessS[ i ] )
33.     {
34.       *padre_1 = i;
35.       found = 1;
36.     }
37.   }
```

**(Continuación en la página siguiente)**

*Listado 2: Implementación del algoritmo de selección de la ruleta.*

```
38.
39.  do
40.    {
41.       // Valor entre 0 y 1
42.       selected = (float)rand() / RAND_MAX;
43.
44.       for ( i = 0, found = 0; i < population_size && found == 0; i++ )
45.       {
46.          if ( selected < fitnessS[ i ] )
47.          {
48.             *padre_2 = i;
49.             found = 1;
50.          }
51.       }
52.    }
53.    while ( *padre_2 == *padre_1 );
54.  }
```

*Listado 3: Implementación del algoritmo de selección de la ruleta.*

En las líneas 30 a 37 se buscará la porción, quesito de la ruleta, que corresponde al valor aleatorio seleccionado. Este valor será el primer padre a devolver por la función.

De la línea 39 a 54, se repetirá la operación anterior para seleccionar el segundo padre, evitando que se seleccione el primer padre.

## *4.2. Herramienta software para predicción*

GetForecastedValues es la herramienta desarrollada para la extracción automática y prueba de los resultados de los experimentos, facilitando el estudio de los mismos y evitando el tedioso trabajo que supone hacer toda la tarea de manera manual.

Hasta el momento, el profesor D. Juan Peralta, realizaba las pruebas y la extracción de los resultados de los experimentos generados de manera manual. Esto implicaba el seguimiento de una serie de pasos que, de no seguir con cuidado, podía llevar a equivocación y a la obtención de resultados no fiables. Estos pasos implicaban la descompresión del experimento a probar previamente, ya que debido al enorme tamaño ocupado por los ficheros generados en cada experimento, se deben comprimir para optimizar el espacio ocupado por cada uno de ellos. Este paso consume una parte importante del tiempo dentro de los pasos. Tras esto se busca la mejor red obtenida en el experimento, observable en el fichero "*last_generation.txt*", y se extrae a un directorio de trabajo. A continuación se genera a mano un fichero de patrones de SNNS. Y se lanza el *shell-script* previamente modificado con los parámetros de la red escogida para obtener los resultados que esa red es capaz de dar. Por último, mediante una sencilla utilidad que se limita a recorrer el fichero de texto en busca de un patrón y extraer lo que tiene en una determinada línea del fichero de resultados obtenido se consiguen los resultados finales para el experimento.

Recorrer todos estos pasos para cada experimento puede llevar una media de 30 minutos.

### 4.2.1. Análisis

En este punto enumeraremos los requisitos funcionales planteados por el profesor que debe cumplir la aplicación. En la figura 12 se pueden observar los casos de uso a cumplir por la utilidad a desarrollar.

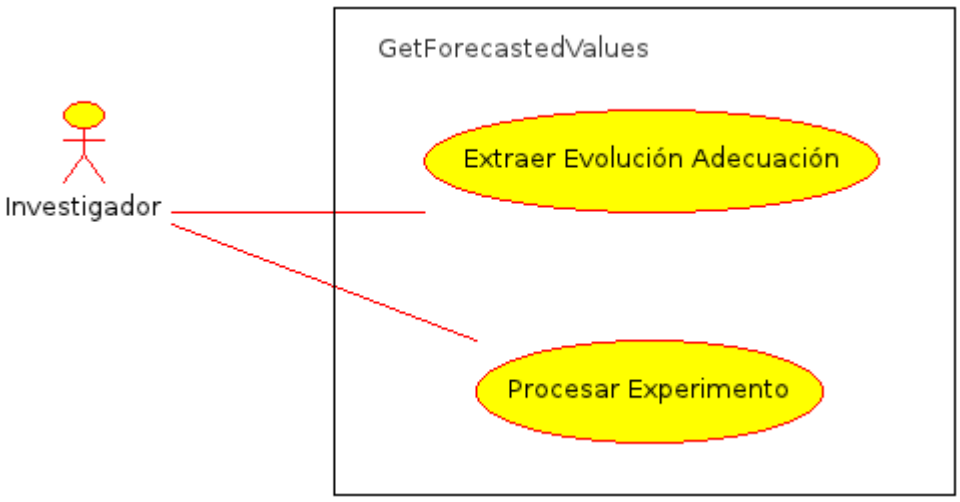


*Figura 12: Casos de uso.*

- REQ01 (Extraer Evolución Adecuación). Se permitirá extraer la evolución de la función de adecuación de cada experimento de forma automática a un fichero de texto "*<fichero_zip>_fitness.res*".
- REQ02 (Procesar Experimento). Se podrá procesar un experimento completo y obtener el resultado de la experimentación en un fichero de texto de resultados "*<fenotipo>.res*".

### 4.2.2. Diseño

Para poder cumplir los requisitos comentados es necesario conocer como es la estructura de ficheros de un experimento generado, que datos obtenidos con el experimento son relevantes a la hora de su evaluación y, por último como se prueba la red generada por el experimento y se extraen las resultados obtenidos por la misma.

Como ya se ha comentado, debido al enorme tamaño que ocupa cada experimento, los experimentos se almacenan comprimidos en formato ZIP de manera que se pueda ahorrar el máximo espacio posible en los dispositivos de almacenamiento. Aún así cada experimento tiende a ocupar desde los 120Mb a los 2,2Gb. Con esta premisa el programa debe ser capaz de leer directamente estos ficheros comprimidos, tratando así de ahorrar tiempo evitando tener que descomprimir cada experimento para tratarlo

La estructura de ficheros de un experimento tiene el aspecto presentado en la figura 13. Lo primero que hay que resaltar es que todas las generaciones del experimento se encuentran dentro de una carpeta, "*nets*" en este ejemplo, pero que puede ser cualquier otra definida por el investigador.

Dentro de esta carpeta están dispuestas y ordenadas todas las generaciones del experimento en su correspondiente carpeta "*generacionXX*", siendo XX el número de generación. En la carpeta de cada generación se pueden ver todas las redes generadas en su correspondiente carpeta, nombradas según los parámetros usados en la red neuronal generada por el AG, y dentro de estas, los ficheros generados por cada red.

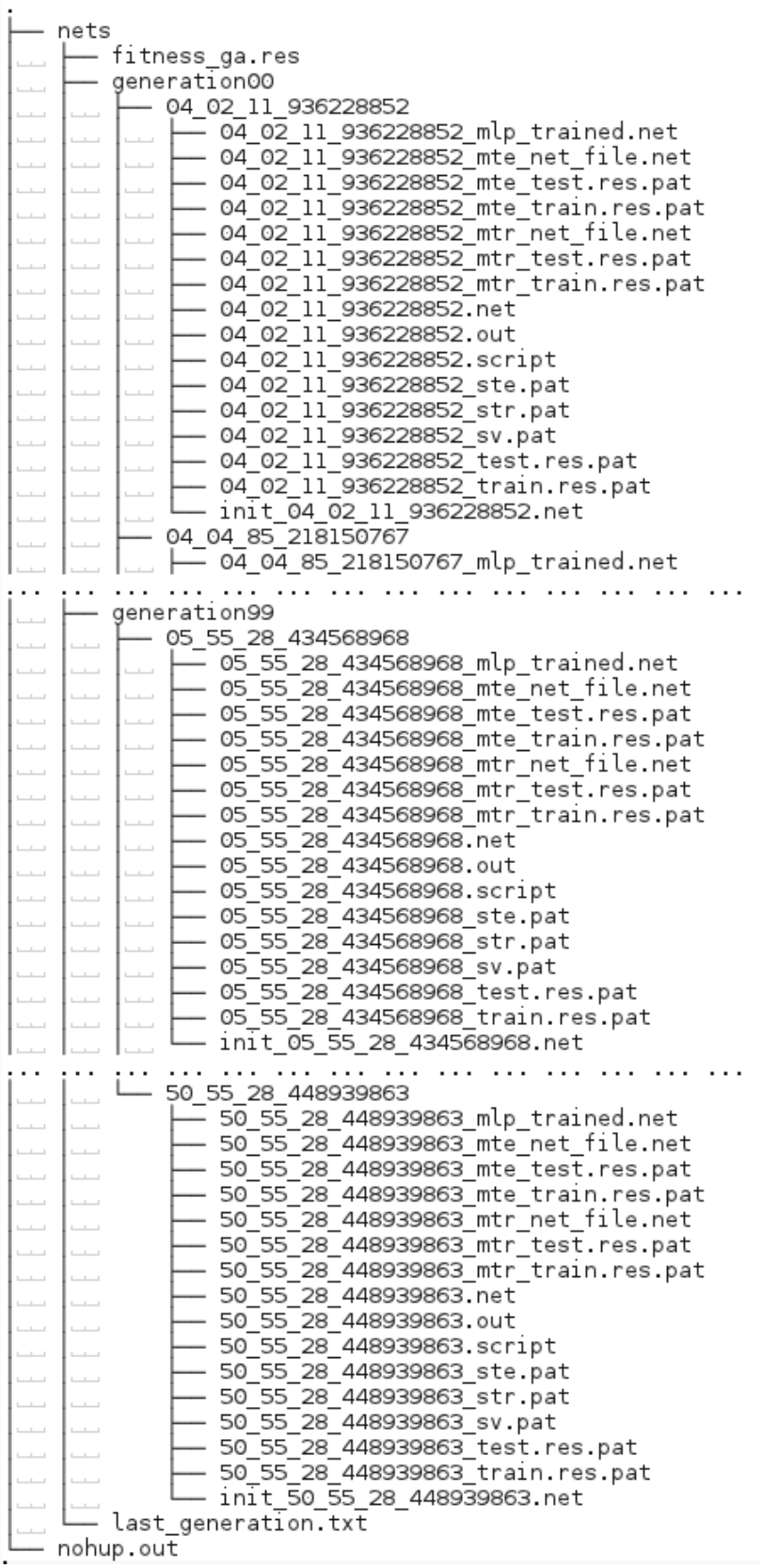


*Figura 13: Estructura de ficheros de un experimento*

De entre estos ficheros generados por cada red en cada generación caben destacar dos:

- <parámetros>_mte_net_file.net. Contiene el guión de SNNS que define la red generada para poder reutilizarla de nuevo.
- *<parámetros>_sv.pat*. Contiene un fichero de SNNS para definir un patrón de la red. Completando este patrón se usará para probar la RNA generada.

Siendo *<parámetros>* los parámetros de la RNA expresados de la siguiente forma:

II_OO_AA_SSSSSSSSS

Donde cada conjunto de letras define:

- II. Número de nodos de entrada.
- OO. Número de capas ocultas.
- AA. Factor de aprendizaje.
- SSSSSSSSS. Semilla usada en la generación.

Además de estos ficheros, dentro de la rama de las generaciones, tenemos otros dos ficheros importantes, estos son:

- last_generation.txt. Este fichero contiene los datos de la última generación del experimento, los cuales podemos suponer mejores que el resto. En concreto el cromosoma 00 contiene la mejor red ya que es la que mejor adecuación ha obtenido.
- fitness_ga.res. Contiene todas las generaciones obtenidas en el experimento.

La estructura de ambos ficheros es la misma, se puede ver en la figura 11, ya que "*last_generation.txt*" tan sólo es la extracción de la última generación contenida en el fichero "*fitness_ga.res*". Tenemos un inicio de sección de generación y la descripción del cromosoma con su fenotipo y función de adecuación obtenida.

```
GENERACION 0
chrom 000: 9 9 6 7 5 8 1 6 4 0 9 9 5 9 4 7 | PHENOTYPE: 50_68_58_1640995948  FITNESS: 0.001024  CONT_DIR: 1
...
GENERACION 1
chrom 000: 9 9 6 7 5 8 1 6 4 0 9 9 5 9 4 7 | PHENOTYPE: 50_68_58_1640995948  FITNESS: 0.001024  CONT_DIR: 0
...
```

Figura 14: Estructura del fichero "fitness_ga.res".

Con estos datos ya podemos diseñar el algoritmo que satisfará el primer requisito de la aplicación.

### *Diseño REQ01*

Según la descripción del requisito 1, se debe de poder obtener la función de adecuación de cada experimento de manera automática, por ello y como ya se ha visto en el apartado anterior, el fichero "*fitness_ga.res*" contiene entre otros datos la función de adecuación de cada generación de cada experimento por cromosoma. Para obtener estos datos se necesitarán dos parámetros el directorio o fichero donde se encuentran los experimentos y la ruta dentro de cada fichero comprimido en formato zip a la carpeta de las generaciones, ya que esta puede variar según el investigador que realice los experimentos. El pseudocódigo inicial tanto para el primer requisito como para el segundo será el mostrado en el listado 4.

```
1. Si el primer parámetro es un directorio.
2.         Por cada fichero comprimido en formato zip del directorio.
3.                 Extraer la función de adecuación/Procesar experimento
4.         Repetir 2.
5. Sino.
6.         Extraer la función de adecuación del fichero indicado/Procesar experimento
```

*Listado 4: Pseudocódigo para el tratamiento de los ficheros comprimidos.*

Ahora, en el listado 5, se verá el pseudocódigo correspondiente al primer requisito, la extracción del la función de adecuación de un fichero.

```
1. Abrir el fichero comprimido.
2. Localizar, según la ruta de las generaciones, y abrir el fichero "fitness_ga.res".
3. Por cada línea del fichero hacer.
4.         Si la línea contiene la cadena "chrom 000:".
5.                 Si la línea contiene la cadena "FITNESS:".
6.                         Partir el resto de la línea en palabras separadas por espacio.
7.                         Almacenar en la cadena de salida la primera palabra.
8. Repetir 3.
9. Devolver y escribir en el fichero "<nombre_fichero_zip>_fitness.res" la cadena de salida.
```

*Listado 5: Pseudocódigo para obtener la evolución de la función de adecuación.*

Siempre se cogerá el cromosoma 000 ya que se supone el mejor de cada generación.

### *Diseño REQ02*

El diseño de este requisito es un poco más complejo, ya que requiere de un mayor número de pasos para ser completado. En el listado 6 se puede ver el pseudocódigo principal del requisito.

Tras el pseudocódigo del listado 6 se explicará con algo más de detalle los puntos que lo contienen.

1. Leer el fichero con la serie temporal normalizada usada en el experimento.
2. Generar la estructura de directorios para procesar el resultado (Nets/Patterns).
3. Leer el fichero "*last_generation.txt*", del que se obtendrá la mejor generación y el fenotipo de esta.
4. Extraer en "*Nets*" el fichero "*<fenotipo>_mte_net_file.net*" obtenido del experimento.
5. Generar a partir del fichero de la serie temporal un fichero patrón de SNNS en "*Patterns*", "*<fenotipo>_sv.pat*", cuyo número de entradas corresponde al resultado de la mejor generación.
6. Generar el shell-script "*lanzar_validacion.sh*" e invocarlo, lo que probará la mejor red obtenida mediante SNNS con los valores de test y obtendrá un resultado de SNNS en "*Patterns/<fenotipo>_sv.pat.res*".
7. Extraer los valores de la predicción del fichero de resultado "*Patterns/<fenotipo>_sv.pat.res*" en "*<fenotipo>.res*".

*Listado 6: Pseudocódigo para procesar y obtener los resultados de un experimento.*

El fichero de texto que se lee en el primer paso, son los datos normalizados correspondientes a la serie usada en el experimento del cual se pretende obtener los resultados, bien sea de la serie *Temperature*, *Passengers*, *Dow-Jones*, *Quebec* o *Mackey-Glass*. Estos ficheros ya están generados y accesibles por la aplicación.

En el paso 2 se genera un directorio con el nombre del fichero zip del experimento y dentro del mismo se generan dos subdirectorios uno "*Nets"* y otro "*Patterns"*. Con esto se define la estructura de directorios necesaria para posteriormente poder lanzar el *shell-script* que a partir del SNNS obtendrá los resultados.

El cromosoma 000 especificado en el fichero "*last_generation.txt*" define la mejor red obtenida en el experimento por lo que esta red se copiará desde el zip a la estructura de directorios del resultado, "*Nets*" en el paso 4.

En el paso 5 se genera el fichero de patrones en la estructura de directorios del resultado, "*Patterns*". En este fichero se definen el número de patrones que contiene, el número de entradas, que coincidirá con el número de entradas marcado en la red obtenida en la mejor generación (cromosoma 000), el número de salidas, siempre 1 en todos los casos y los valores asociados a las entradas y a la salida.

En el punto 6, la aplicación genera un *shell-script* cuyo ejemplo de código se puede

ver en el listado 7.

Para terminar en el punto número 7 se escriben en el fichero "*<fenotipo>.res*", el conjunto de resultados extraídos del fichero "*Patterns/<fenotipo>_sv.pat.res*".

En este listado 7 la aplicación especifica en los puntos 1, 2 y 3 las rutas a los correspondientes directorios de trabajo del resultado.

En el punto 5 se especifica el directorio donde se encuentran las topologías de red generadas.

En la línea 9, se especifica el número de valores a predecir.

En la línea 11 se invoca al *shell-script* que llamará a los programas de SNNS que entrenarán la red y devolverán los resultados obtenidos para la misma. Se muestra el código en el listado 8.

```
1.  Time_Serie=Mackey_50_100_10
2.  Nets_dir=$Time_Serie/Nets
3.  Patterns_dir=$Time_Serie/Patterns
4.
5.  for topology in '11_165_91_1562142939'
6.  do
7.          Net_file=$Nets_dir/$topology'_mte_net_file.net'
8.          Pattern_file=$Patterns_dir/$topology'_sv.pat'
9.          nv4f=56
10.
11.         ./get_forecast_values.sh $Net_file $Pattern_file $Pattern_file.res $nv4f
12. done
13.
14. touch 11_165_91_1562142939.end
```

*Listado 7: Shell-script ejemplo "lanzar_Validation.sh".*

Por último, en la línea 14 se crea un fichero vacío, el cual actúa como un semáforo para el programa *GetForecastedValues* en Java, que le indicará que se han terminado de generar los resultados por parte del SNNS.

En el listado 8 se puede ver el código del *shell-script* "*get_forecast_values.sh*" que será el encargado de llamar a los programas de SNNS para probar la red y obtener los resultados de su predicción.

Este *shell-script* espera 4 parámetros:

- $1. El fichero de la red.
- $2. El fichero de patrones.
- $3. El fichero donde devolver los resultados.
- $4. Número de valores a predecir.

En el anexo 8.2 se puede ver el código del programa para "*batchman*" intérprete *batch* de SNNS que generan los *shell-scripts* auxiliares en las líneas 6 a 14. Este programa carga los ficheros indicados y llama a las funciones de SNNS que se encargan de probar la red y devolver los resultados.

En el bucle de las líneas 18 a 32 se invoca a SNNS para probar la red actual, luego se van generando los sucesivos patrones necesarios mediante "*get_next_pat_2_forecast*" para probar a predecir el siguiente valor con la red hasta probar todos los valores a predecir.

Una vez terminada la ejecución de la aplicación se podrá operar con los resultados obtenidos según convenga al investigador. El tiempo de proceso aproximado que consume la aplicación por experimento procesado es de una media inferior a 30 segundos frente a los 30 minutos de media que podía llevar el proceso manual y sin el riesgo de confusiones.

```
1.  net_file=$1
2.  pattern_file=$2
3.  result_file=$3
4.  nv2f=$4
5.
6.  ./create_batch_2_forecast_part-1.sh $net_file $pattern_file $result_file >
    batch_2_forecast_part-1.bat
7.
8.  mv batch_2_forecast_part-1.bat Program_batchman
9.
10. cd Program_batchman
11.
12. cat batch_2_forecast_part-1.bat batch_2_forecast_part-2.bat > batch_2_forecast.bat
13.
14. mv batch_2_forecast.bat ../
15. cd ..
16. cp $pattern_file $pattern_file.original
17.
18. for (( ind=1 ; ind <=$nv2f ; ind=ind+1));do
19.
20.   batchman -q -f batch_2_forecast.bat -l forecast_log.out
21.
22.   cp $result_file $result_file-$ind.res
23.
24.   pwd
25.
26.   echo ./get_next_pat_2_forecast -f $result_file -c `expr $ind` -o $pattern_file.$ind.pat
27.
28.   ./get_next_pat_2_forecast      -f $result_file -c `expr $ind` -o $pattern_file.$ind.pat
29.
30.   cp $pattern_file.$ind.pat $pattern_file
31.
32. done
```

*Listado 8: Shell-script "get_forecast_values.sh".*

# 5 EXPERIMENTOS REALIZADOS Y RESULTADOS

En este apartado se van a documentar los experimentos realizados con los cambios ya mencionados en el documento y otros basados en la modificación de los parámetros del AG.

## *5.1. Series temporales*

Para el desarrollo de los experimentos se han seleccionado cinco series temporales que se denominan Passengers, Dow-Jones, Temperature, Quebec y Mackey-Glass. En base a estas series se pueden evaluar los experimentos.

1. Passengers. Esta serie temporal recopila información sobre el número de pasajeros, en miles, de una línea de vuelos internacional medida desde enero de 1949 hasta diciembre de 1960, la fuente es Box & Jenkins (1976).
2. Temperature. Esta otra muestra la temperatura media mensual en el Castillo de Nottingham desde 1920 hasta 1939, la fuente de estos datos es O.D. Anderson (1976).
3. Dow-Jones. Registra los cierres mensuales del índice Dow-Jones desde agosto de 1968 a agosto de 1981, provienen de Hipel y McLeod ( 1994).
4. Quebec. Representa el número de nacimientos diarios medidos en la ciudad de Quebec desde el 1 de enero de 1977 hasta el 31 de diciembre de 1978.
5. Mackey-Glass. Esta serie se basa en la ecuación diferencial de Mackey-Glass y esta considerada como un punto de referencia para comparar la capacidad de generalización de los diferentes métodos. Esta serie es una serie temporal caótica generada a partir de ecuaciones diferenciales ordinarias con un retardo temporal.

A continuación se va a mostrar una gráfica de cada serie temporal utilizada en la experimentación. En las figuras 15, 16, y 17 se puede ver una representación gráfica de las serie Passengers, Temperature y Dow-Jones respectivamente. Las series Quebec y Mackey-Glass se pueden ver representadas gráficamente en las figuras 18 y 19 respectivamente.

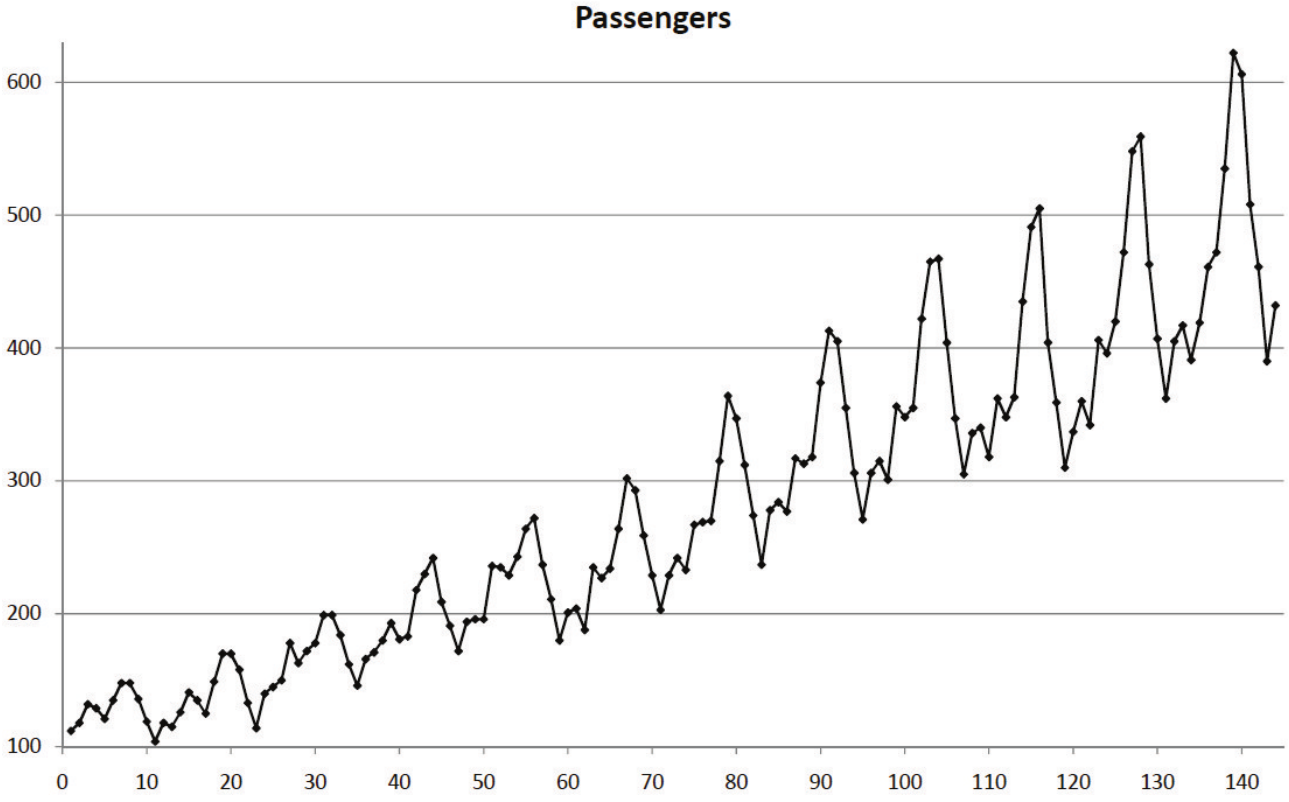


*Figura 15: Serie temporal Passengers.*

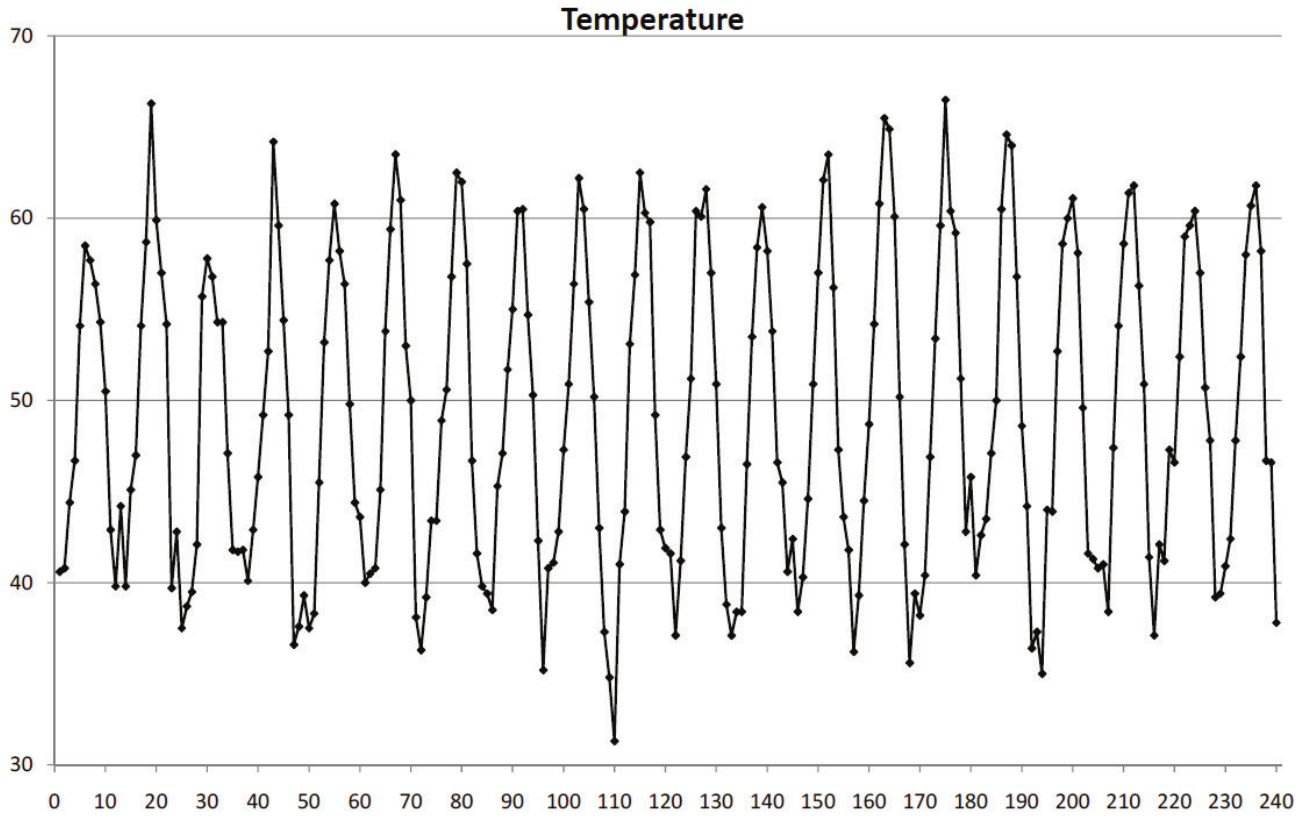


*Figura 16: Serie temporal Temperature.*

A excepción de la serie temporal Mackey–Glass, el resto de series temporales están formadas por datos obtenidos del mundo real, esto hace que su predicción sea algo aún más complicado de por si ya que puede verse influenciada por multitud de factores externos.

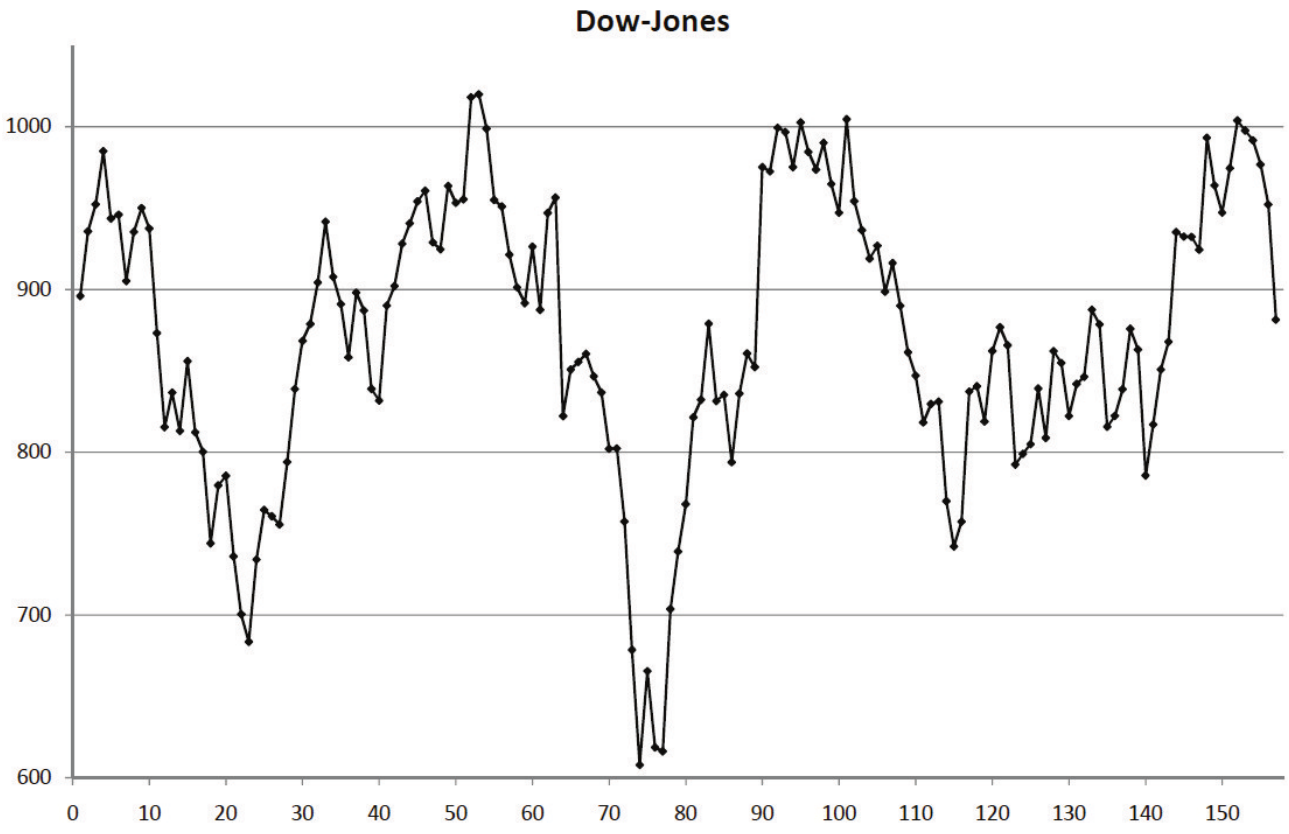


*Figura 17: Serie temporal Dow Jones.*

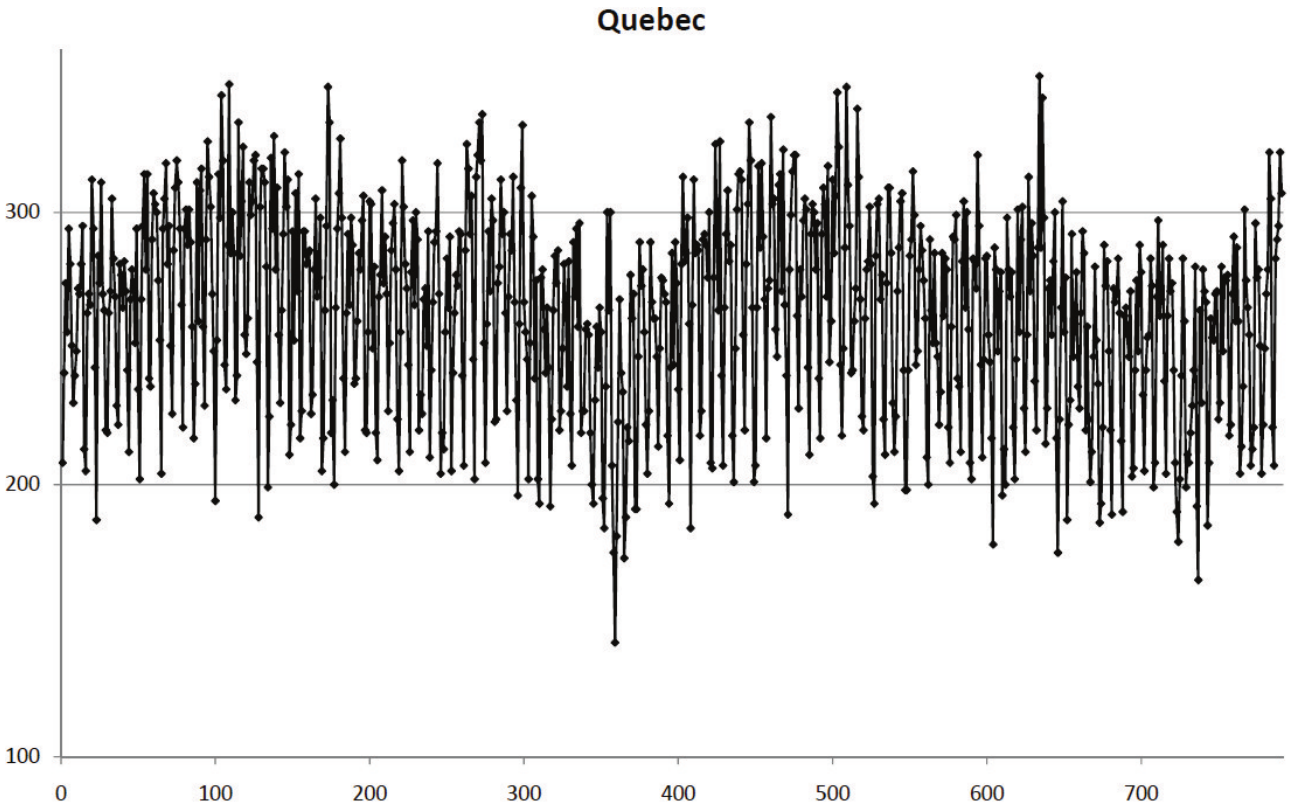


*Figura 18: Serie temporal Quebec.*

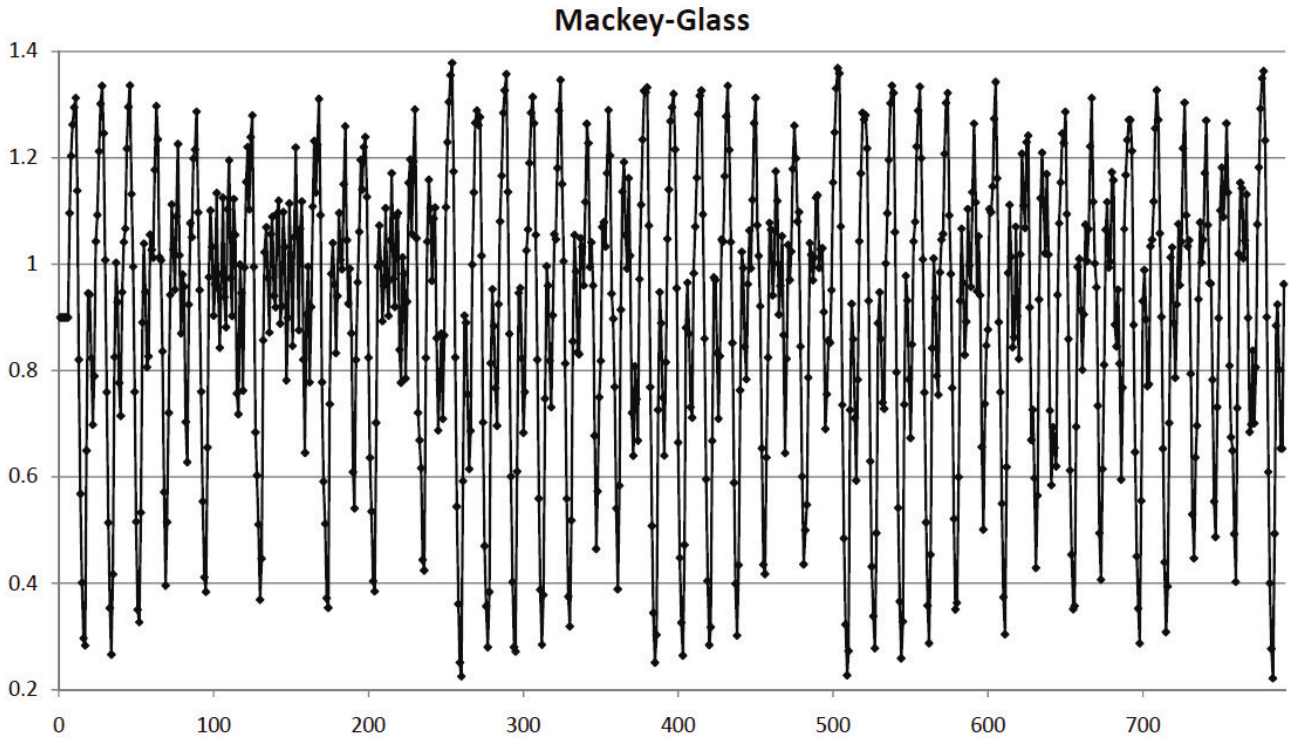


*Figura 19: Serie Temporal Mackey - Glass.*

Cada serie temporal se divide inicialmente en dos subconjuntos, el primero de ellos se compondrá con los datos que se usan para construir el modelo de predicción, mientras que el segundo, denominado conjunto de test, se usará para evaluar las predicciones del modelo mediante los valores más recientes de la serie temporal, a través de los indicadores de error que veremos más adelante. El primer subconjunto se divide a su vez en otros dos subconjuntos el que forma los patrones de entrenamiento y el de validación que se usan en el modelo ADANN.

Según las competiciones NN3 o NN5, en las series temporales largas, más de 700 elementos, como Quebec o Mackey-Glass, hay que predecir los 56 valores siguientes mientras que en series más cortas como Temperature, Dow-Jones o Passengers, basta con los 19 valores siguientes para la generación de un modelo.

Para evaluar lo bueno que es un modelo de predicción se usan medidas de precisión, como el Error Cuadrático Medio (del inglés MSE) o como el Error Simétrico

Porcentual Absoluto (del inglés SMAPE). Estos errores se muestran en las ecuaciones 3 y 4.

$$MSE = \frac{1}{h} \sum_{t=T+1}^{T+h} (e_t^2)$$

*Ecuación 3: Error Cuadrático Medio*

$$SMAPE = \frac{1}{h} \sum_{t=T+1}^{T+h} \frac{|e_t|}{(|y_t| + |\hat{y}_t|)/2} \times 100\%$$

*Ecuación 4: Error Simétrico Porcentual Absoluto*

Donde $e_t = Y_t + \hat{Y}_t$ , $T$ es el instante actual en el que nos encontramos y $h$ es el horizonte a predecir o el número de predicciones que se van a llevar a cabo. La variable $y_t$ es el valor real de la serie e $\hat{y}_t$ es el valor predicho.

Los valores obtenidos con estas ecuaciones son mejores cuanto más pequeños sean. El valor MSE se usa ampliamente en la evaluación de la predicción de series temporales. El valor SMAPE tiene la ventaja de ser independiente de la escala usada, la versión usada es una variante que evita los valores negativos. En las competiciones NN3, NN5 y NNGC1 de predicción de series temporales se usa el valor SMAPE como error. A la hora de evaluar los resultados obtenidos en esta memoria se utilizarán los dos valores como referencia.

## *5.2. Preparación de los experimentos*

Antes de poder lanzar ningún experimento es necesario preparar los valores de las series temporales que se van a utilizar. Para ello se necesita re-escalar los valores dentro del intervalo [0..1], no sólo los valores conocidos sino que también se necesita re-escalar los valores futuros, aquellos que se van a predecir.

Por esto, los límites máximo y mínimo para normalizar no se pueden ajustar a los

valores máximo y mínimo conocidos de la serie temporal Hay que establecer un margen desde el valor máximo y mínimo conocidos por si los valores que se van a predecir fueran mayores o menores que los ya conocidos. Este margen dependerá del parámetro definido por el usuario "*Prct_inc*". En aquellos casos en los que la serie temporal sea estacionaria un valor del 10% para "*Prct_inc*" será suficiente, pero si las series temporales se incrementan o decrementan este valor debería de ser un 50%. Como podría haber valores que se incrementarán después de haberse decrementado, se necesita un margen bastante grande para que los nuevos valores, obtenidos a la salida de la RNA, puedan estar contenidos en el rango [0..1]. Las siguientes ecuaciones, ecuación 5, muestran como obtener los máximos y mínimos normalizados.

$$max4norm = max + (Prct_{inc} \cdot (max - min))$$
$$min4norm = min - (Prct_{inc} \cdot (max - min))$$

*Ecuación 5: Ecuaciones para obtener los valores máximo y mínimo normalizados*

## *5.3. Resultados obtenidos*

Como ya se ha comentado se han usado cinco tipos de series temporales para la elaboración de los experimentos, tres series cortas Dow-Jones, Passengers y Temperature, y dos largas Mackey-Glass y Quebec. Cada experimento se ha lanzado 10 veces por cada una de las series temporales.

A continuación se va ha hacer un estudio sobre como evoluciona la función de adecuación por cada serie temporal y por cada experimento lanzado. En cada gráfica se puede observar como el valor de adecuación va disminuyendo a medida que las generaciones aumentan para llegar a un resultado óptimo. Cuanto más abrupto es el descenso de la gráfica más rápidamente se acerca un resultado.

Para no llenar la memoria con demasiadas gráficas se ha usado la mediana para unificar las gráficas de las diez veces que se ha ejecutado cada experimento por cada serie temporal y, aunque los resultados no sean demasiado representativos, se puede observar como evoluciona la función de adecuación.

Se considera que la mediana es el dato que esta justo en el centro de una serie de números, ordenados en orden creciente o decreciente, que representan la muestra. En el caso de que el número de datos sea par se suman los dos números del medio y se dividen por dos.

Cada gráfica representada se compara con el experimento base algoritmo de selección torneo de 2 padres con 20 individuos como población y 100 generaciones para tener una referencia común.

En las figuras 20, 21, 22, 23 y 24, se representa la evolución de la función de adecuación en los experimentos de selección por torneo de 2 padres con 20 individuos y 100 generaciones frente a los de selección por torneo de 2 padres con 50 individuos y 100 generaciones.

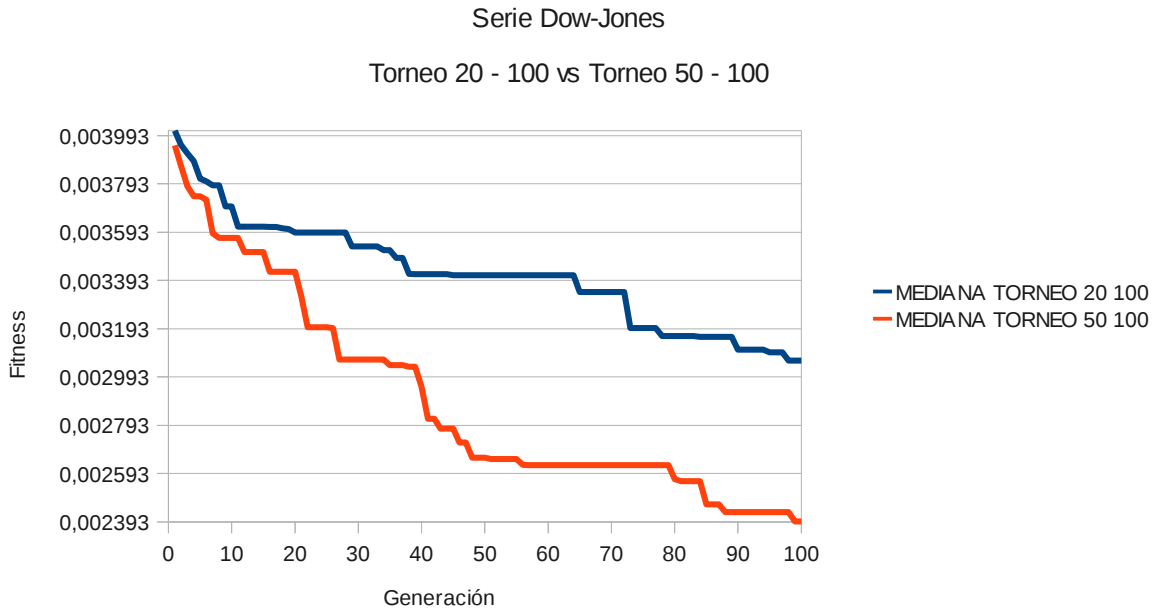


*Figura 20: Gráfica de evolución de la fitness de Dow-Jones. Torneo 50 individuos.*

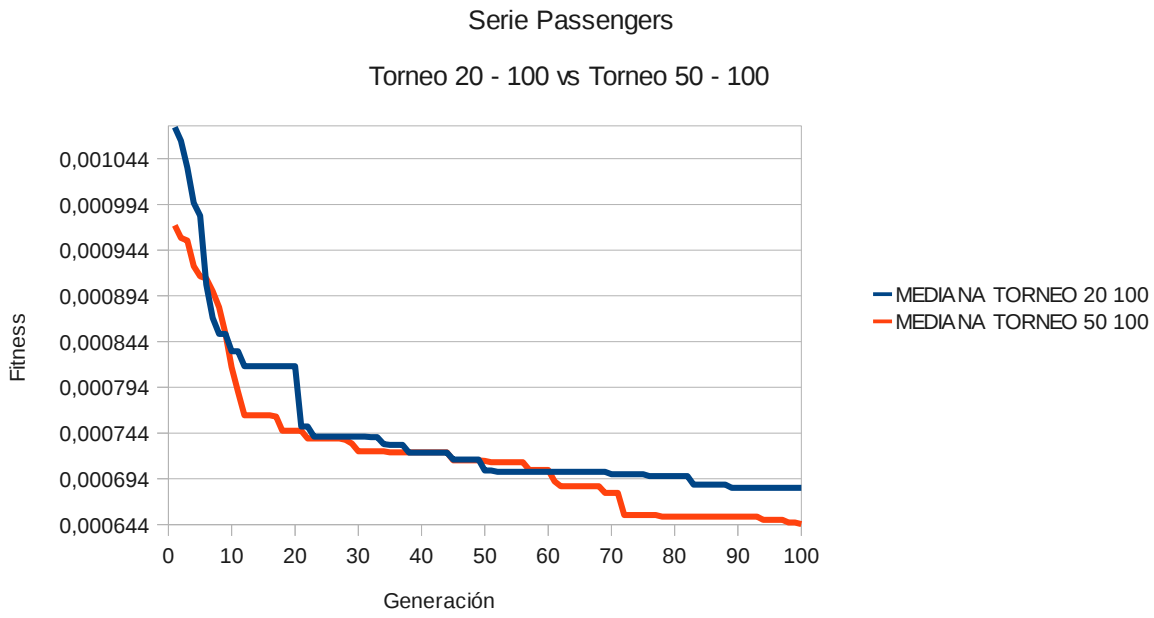


*Figura 21: Gráfica de evolución de la fitness de Passengers. Torneo 50 individuos.*

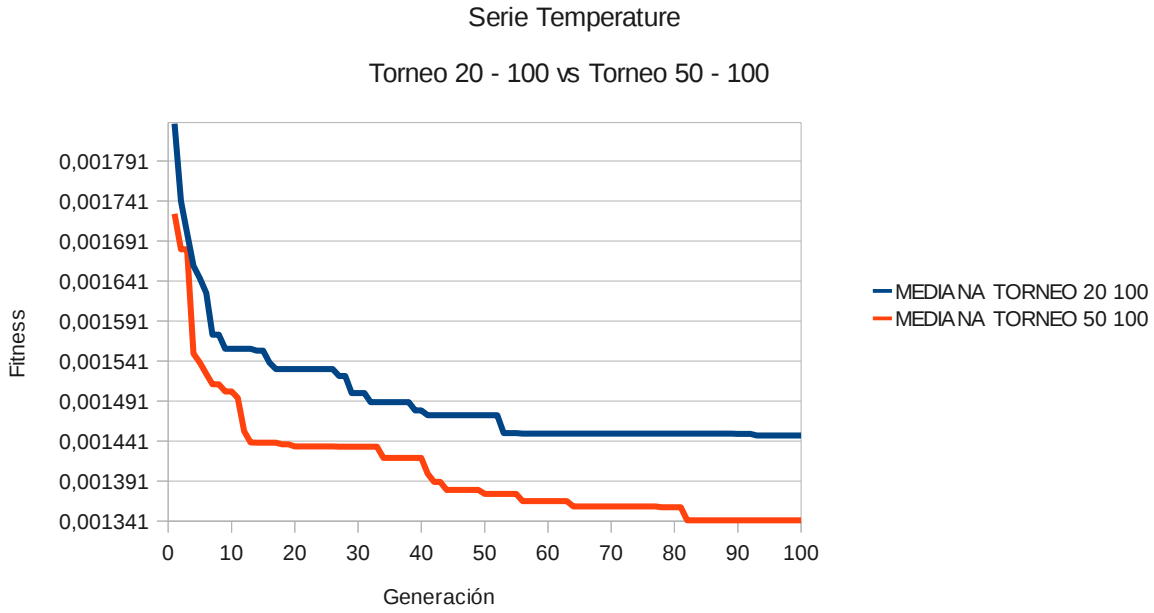


*Figura 22: Gráfica de evolución de la fitness de Temperature. Torneo 50 individuos.*

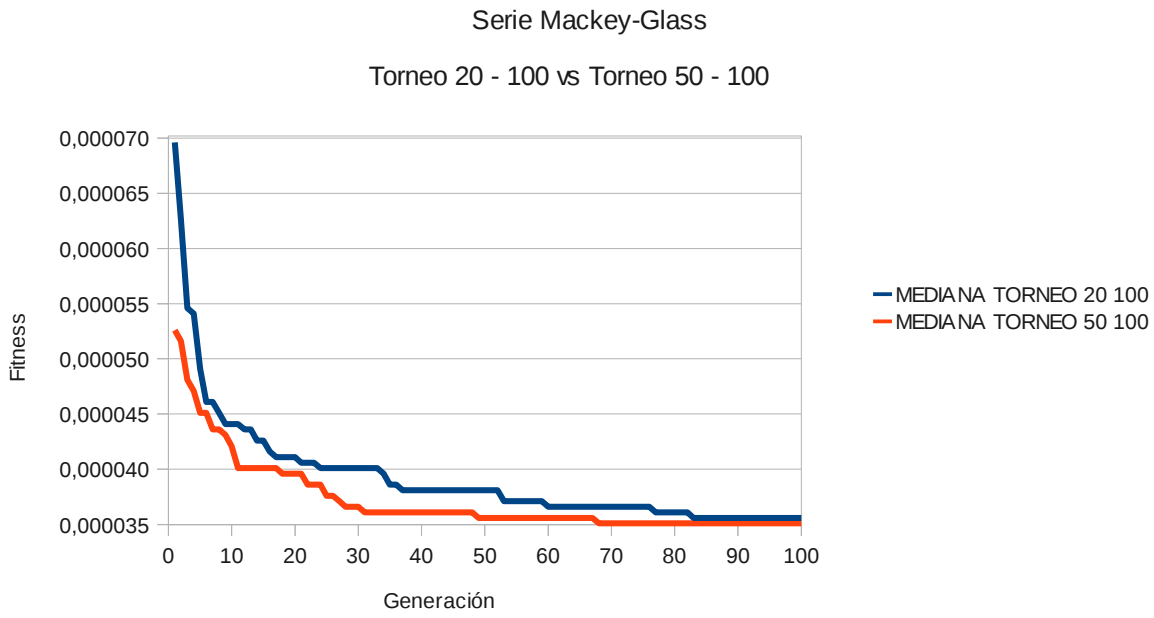


*Figura 23: Gráfica de evolución de la fitness de Mackey-Glass. Torneo 50 individuos.*

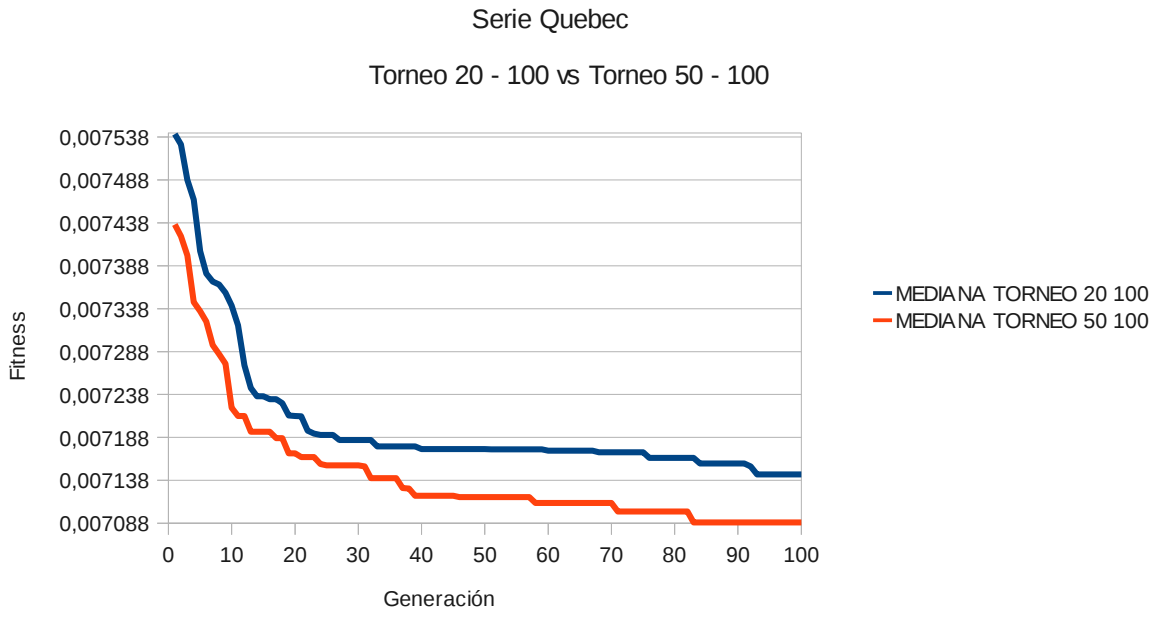


*Figura 24: Gráfica de evolución de la fitness de Quebec. Torneo 50 individuos.*

Como se puede observar en las gráficas mostradas la línea de evolución de la adecuación para el torneo de 2 padres con 50 individuos siempre es mejor con respecto al torneo de 2 padres con 20 individuos a pesar de que, como ya se verá más adelante, los mejores valores obtenidos para la función de adecuación se obtengan en el torneo de 2 padres con 20 individuos. Este hecho ha sido ya tratado y estudiado en [7], donde se plantea que el valor de adecuación es difícil de medir de manera precisa y exacta. A pesar de que el valor dado por la mejor RNA de una generación represente que dicho individuo es mejor, en la práctica, al llevar a cabo la tarea de predicción no suele ser así. El segundo o tercer individuo de esa generación puede obtener mejores resultados en el futuro aunque nosotros no podamos saberlo de antemano. Lo mismo ocurre en nuestro caso, donde aunque nos aseguramos un buen valor de adecuación, comparando dichos valores de dos métodos diferentes no es suficiente para saber cual es mejor, sería necesario llevar a cabo sus respectivas predicciones para poder afirmarlo.

En todas las gráficas se puede ver una rápida evolución dentro de las 20 primeras generaciones, a excepción de la serie Dow-Jones donde la evolución para el torneo de 2 padres con 50 individuos tiene una evolución más progresiva a lo largo de las 50 primeras generaciones mientras que la evolución en el torneo de 2 padres con 20 individuos es incluso más progresivo

En las figuras 25, 26, 27, 28 y 29 se va a ver la evolución de la función de adecuación para los experimentos del torneo de 2 padres con 20 individuos y 100 generaciones frente al torneo de 2 padres con 20 individuos y 250 generaciones. Se puede observar como evoluciona la función de adecuación a partir de la generación 100 para ver si la evolución compensa el coste de computación adicional.

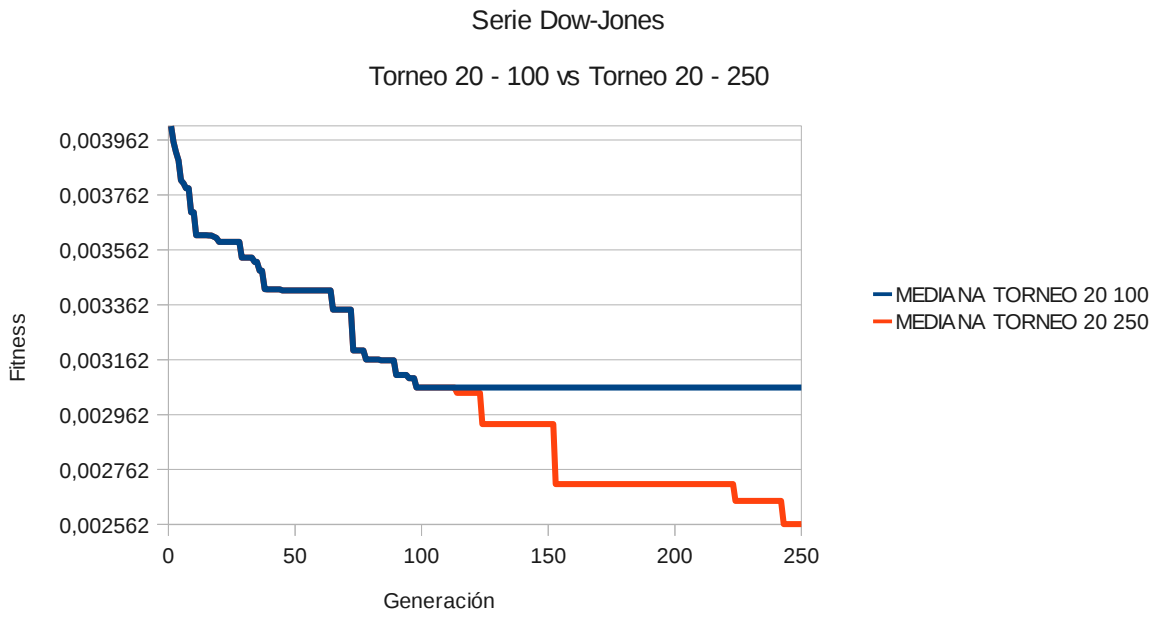


*Figura 25: Gráfica de evolución de la fitness de Dow-Jones. Torneo 20 individuos.*

En la figura 25, la evolución de la adecuación para la serie Dow-Jones es la más destacada del conjunto de experimentos con 250 generaciones ya que, a diferencia del resto, la mejora en la evolución de la fitness a partir de la generación 100 es bastante importante, casi igual que en el primer intervalo de generaciones.

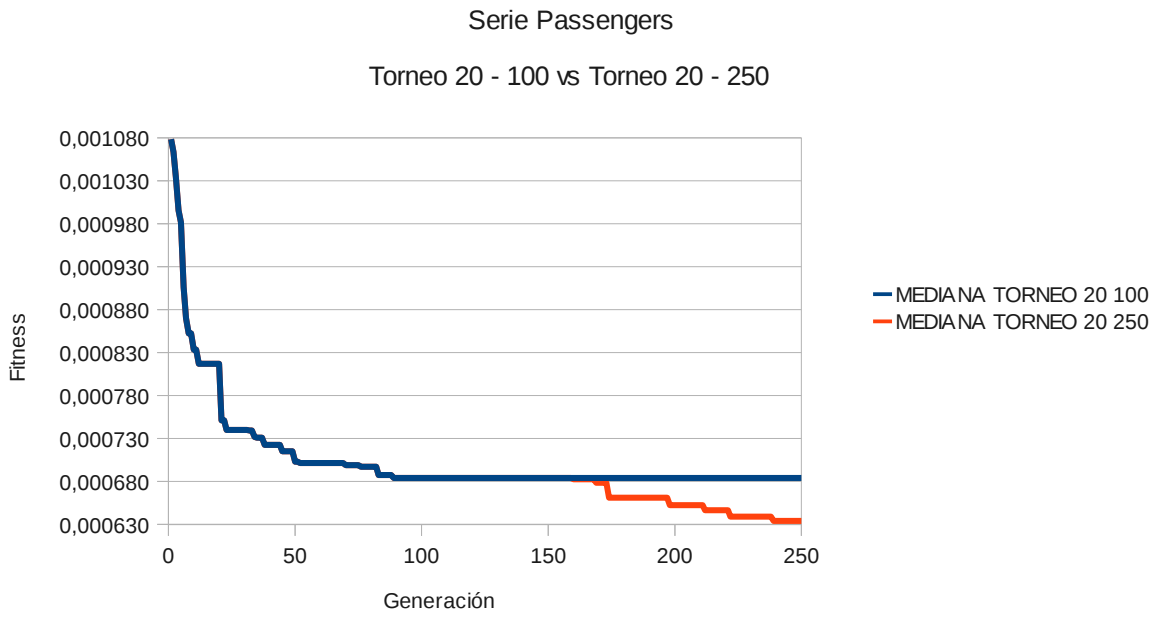


*Figura 26: Gráfica de evolución de la fitness de Passengers. Torneo 20 individuos.*

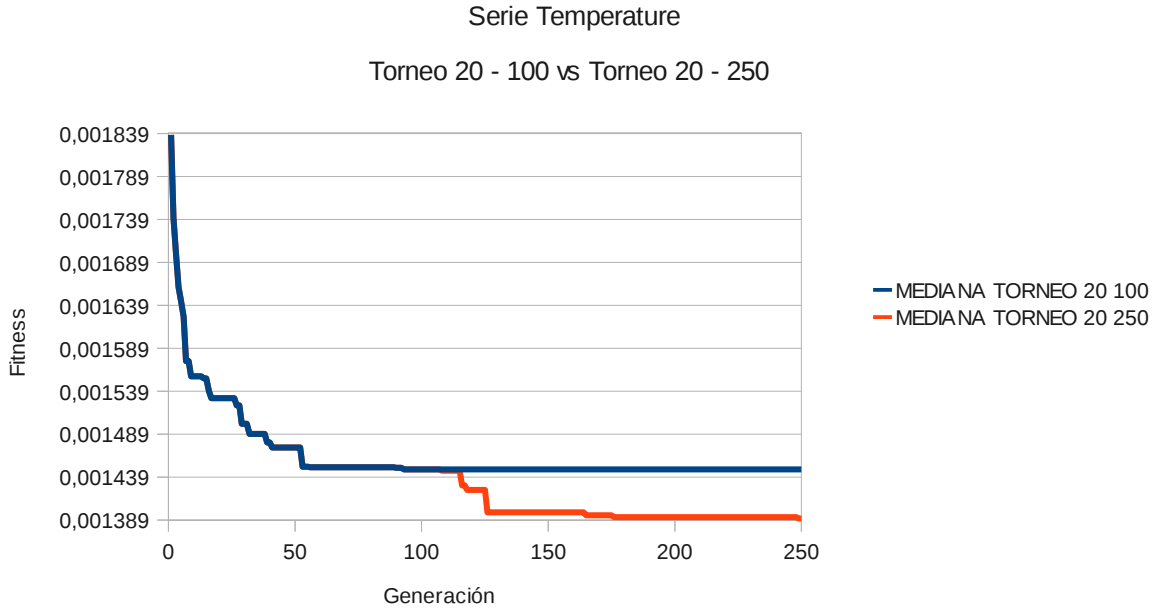


*Figura 27: Gráfica de evolución de la fitness de Temperature. Torneo 20 individuos.*

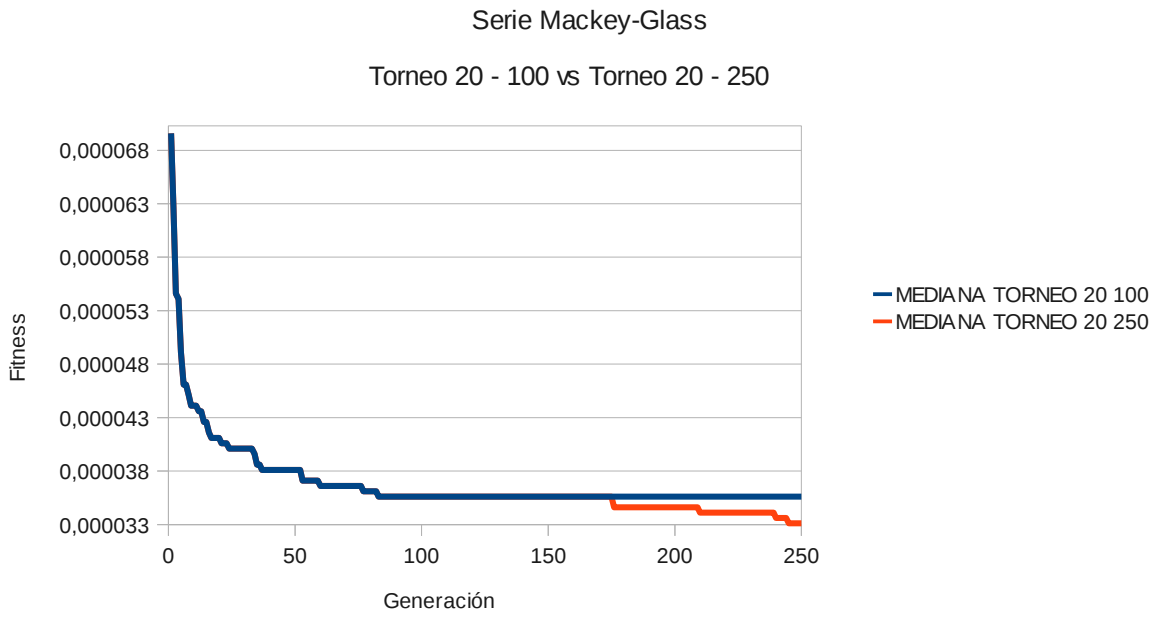


*Figura 28: Gráfica de evolución de la fitness de Mackey-Glass. Torneo 20 individuos.*

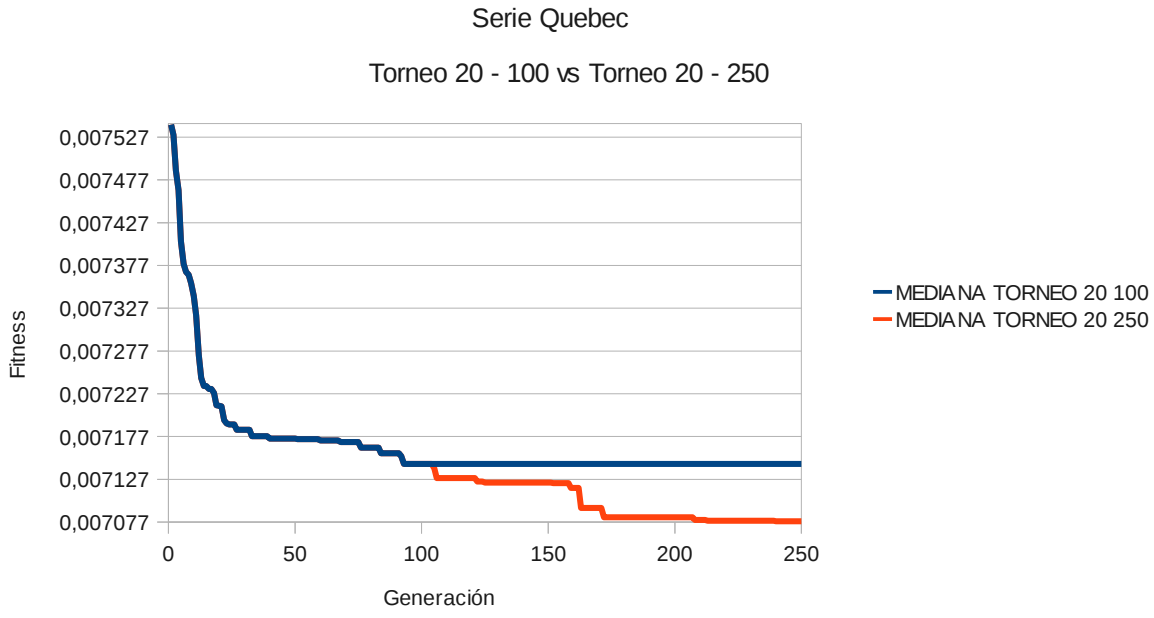


*Figura 29: Gráfica de evolución de la fitness de Quebec. Torneo 20 individuos.*

En el resto de figuras, 26, 27, 28 y 29 de los experimentos para 250 generaciones se ve, claramente que la mejora obtenida a partir de la generación 100 no justifica el

costo en computo que conlleva.

También se puede observar que las figuras 26, 27, 28 y 29 tienen una importante evolución en las primeras 50 generaciones, a partir de las cuales la evolución se suaviza, mientras que en la figura 20, la evolución se mantiene más o menos proporcional a lo largo de toda la gráfica.

En las figuras 30, 31, 32, 33 y 34, se puede apreciar la evolución de la función de adecuación para los experimentos del torneo de 2 padres con 20 individuos y 100 generaciones contra el torneo de 3 padres con 20 individuos y 100 generaciones.

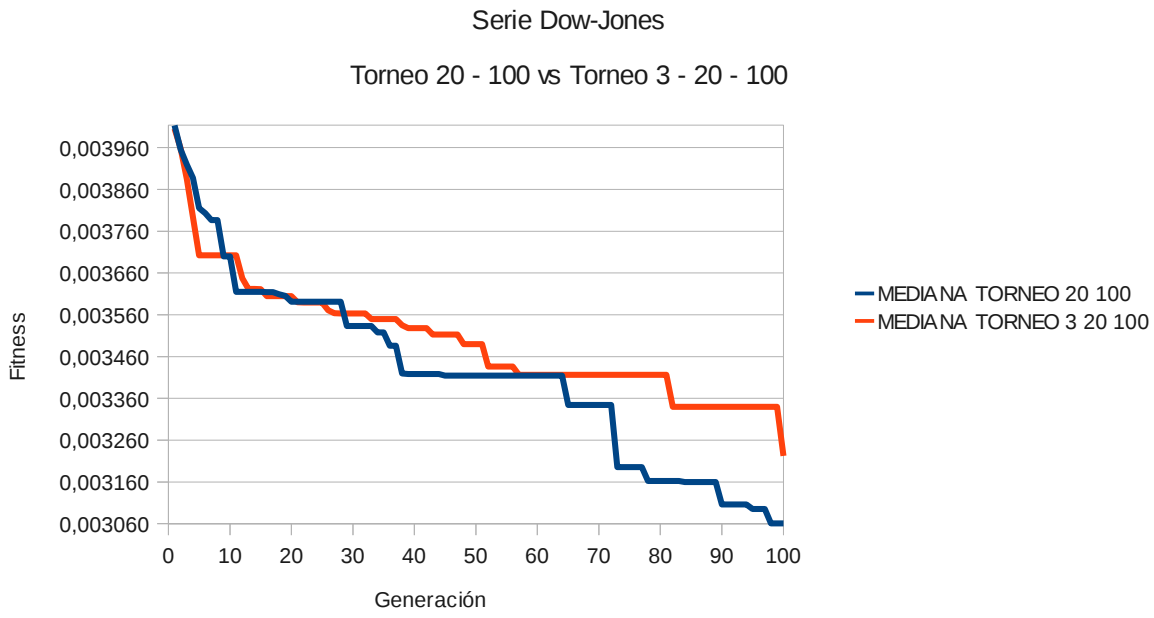


*Figura 30: Gráfica de evolución de la fitness de Dow-Jones. Torneo 3 padres 20 individuos.*

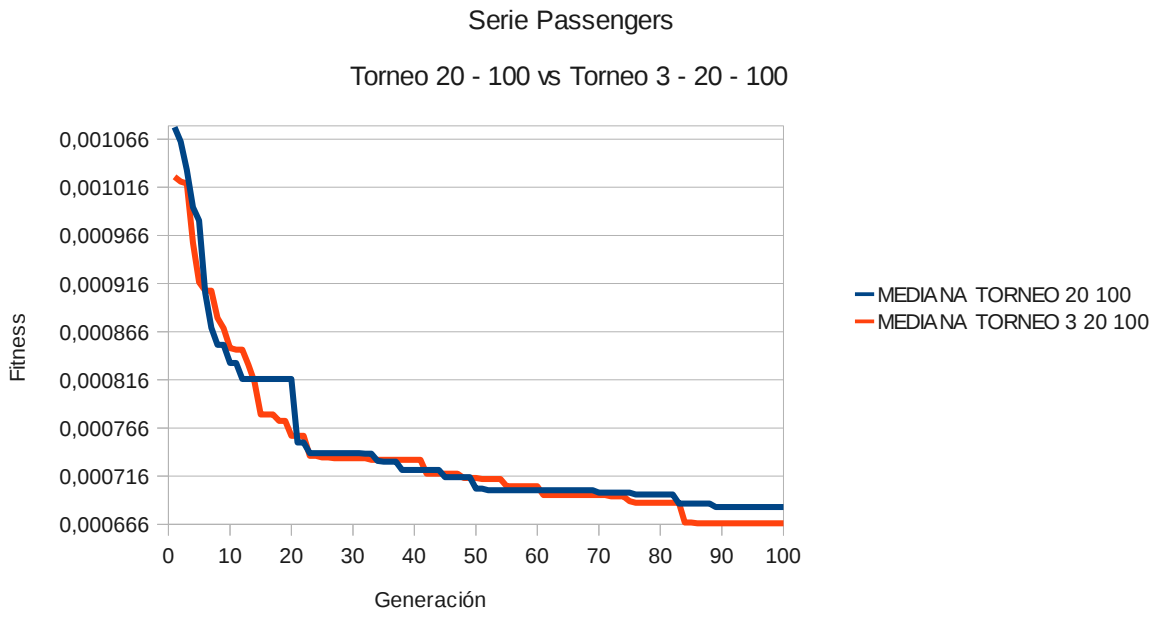


*Figura 31: Gráfica de evolución de la fitness de Passengers. Torneo 3 padres 20 individuos.*

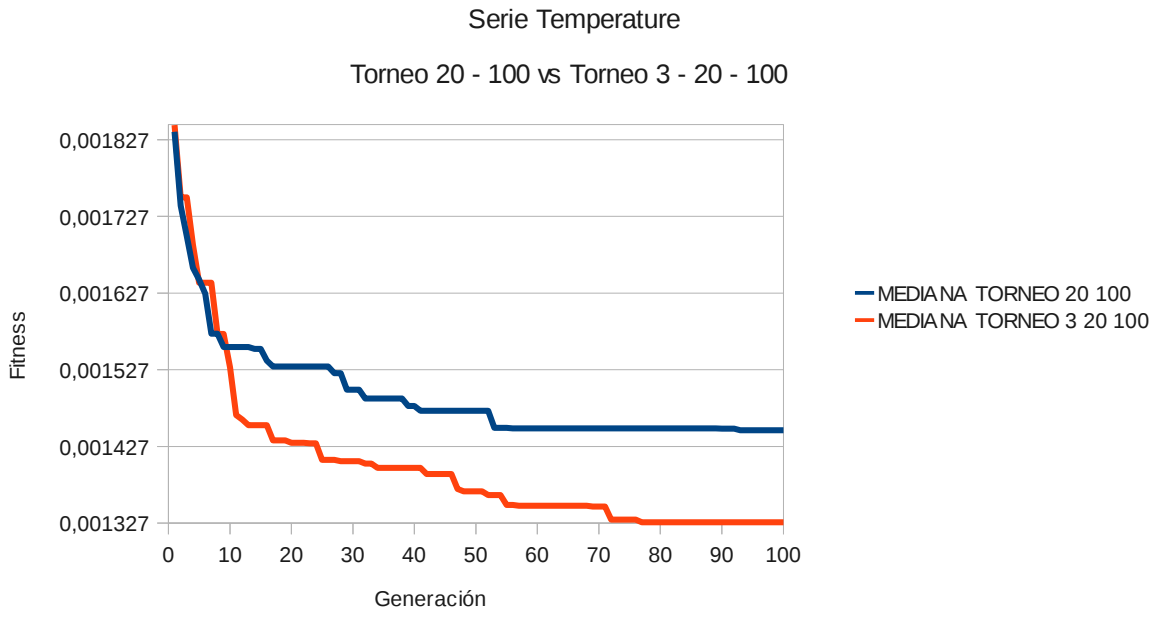


*Figura 32: Gráfica de evolución de la fitness de Temperature. Torneo 3 padres 20 individuos.*

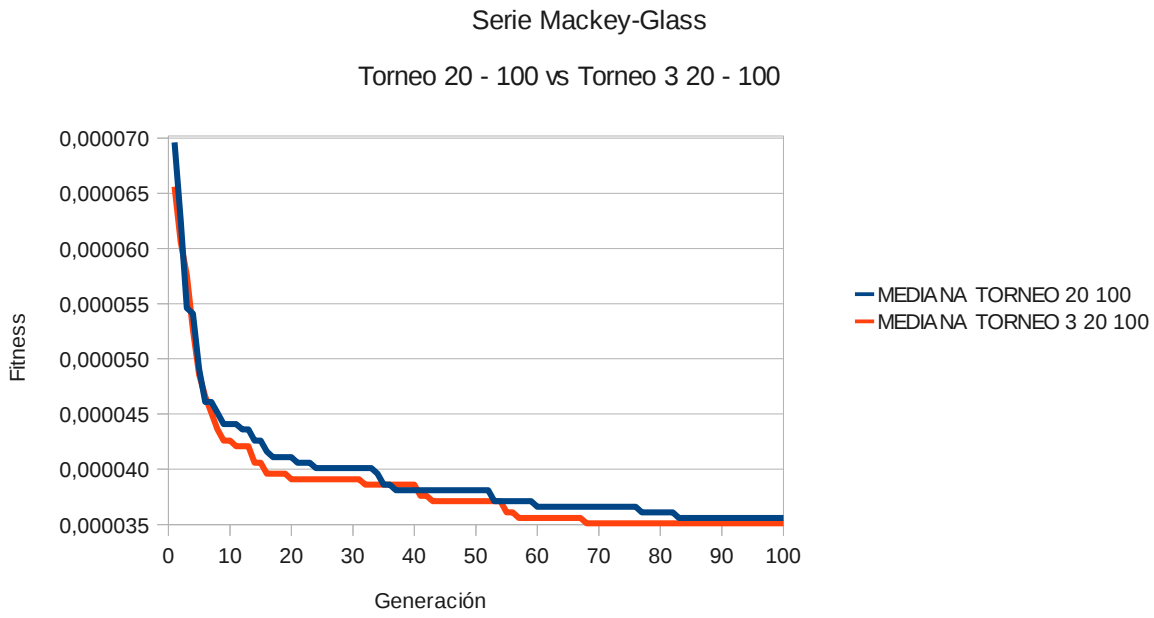


*Figura 33: Gráfica de evolución de la fitness de Mackey-Glass. Torneo 3 padres 20 individuos.*

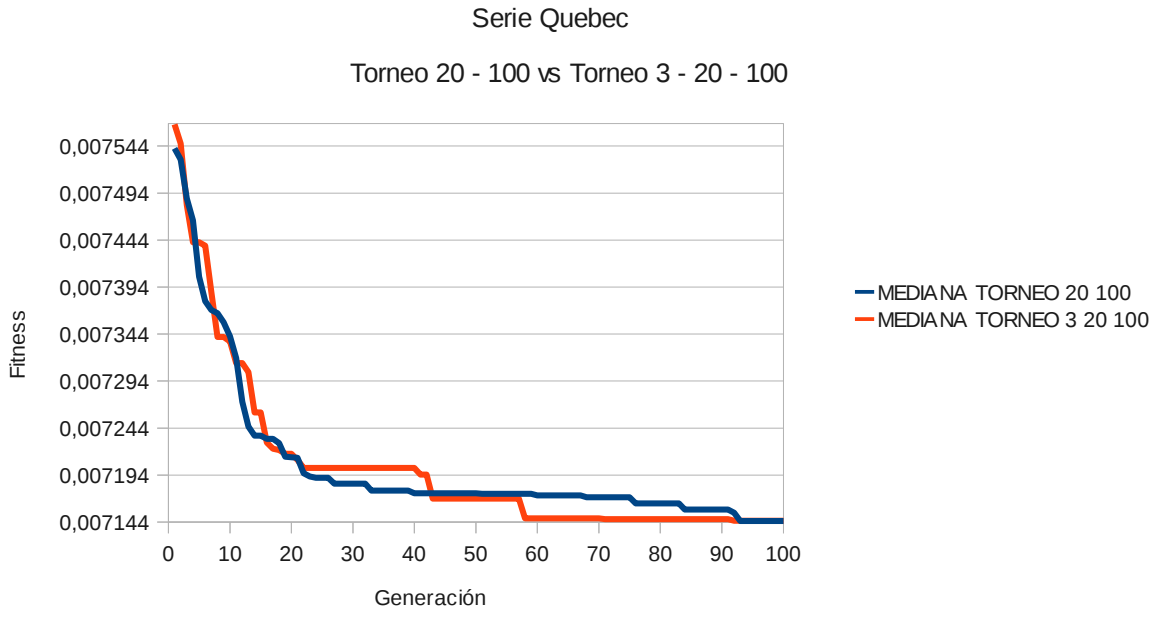


*Figura 34: Gráfica de evolución de la fitness de Quebec. Torneo 3 padres 20 individuos.*

Como ocurría en el caso anterior en los experimentos del torneo de 3 padres, la evolución de la fitness es muy sobresaliente en las primeras 20 generaciones en las

figuras 31, 32, 33 y 34, mientras que en la figura 30, serie Dow-Jones, la evolución es proporcional a lo largo de toda la gráfica.

Comparada la evolución de los experimentos para el torneo de 3 padres con respecto al torneo de 2 padres, se puede adivinar que las curvas son muy aproximadas por lo que no parece que aporte ninguna ventaja sobre el experimento de referencia.

En las figuras 35, 36, 37, 38, y 39 se tiene la evolución de la función de adecuación para los experimentos del torneo de 2 padres con 20 individuos y 100 generaciones frente a los experimentos cuyo algoritmo de selección es el de la ruleta con 20 individuos y 100 generaciones.

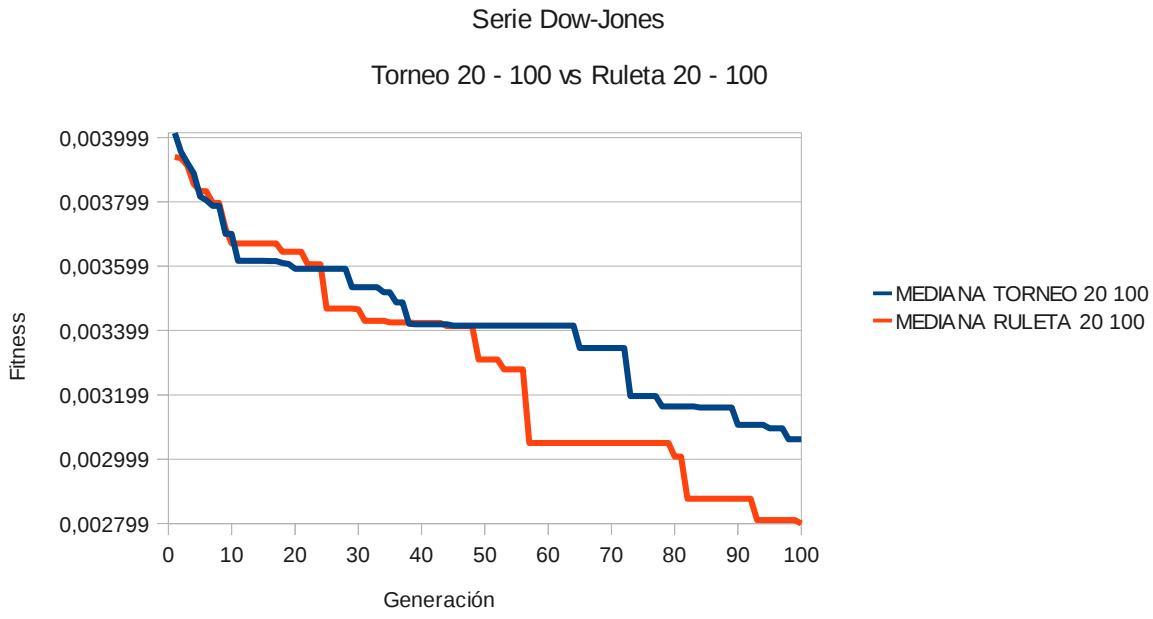


*Figura 35: Gráfica de evolución de la fitness de Dow-Jones. Ruleta 20 individuos.*

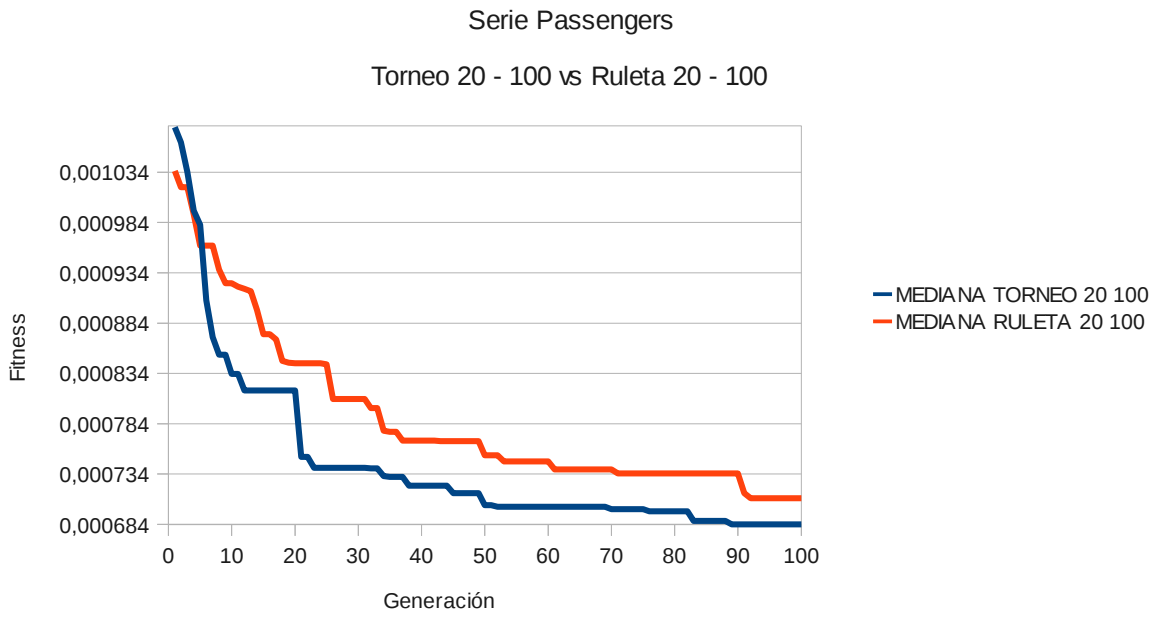


*Figura 36: Gráfica de evolución de la fitness de Passengers. Ruleta 20 individuos.*

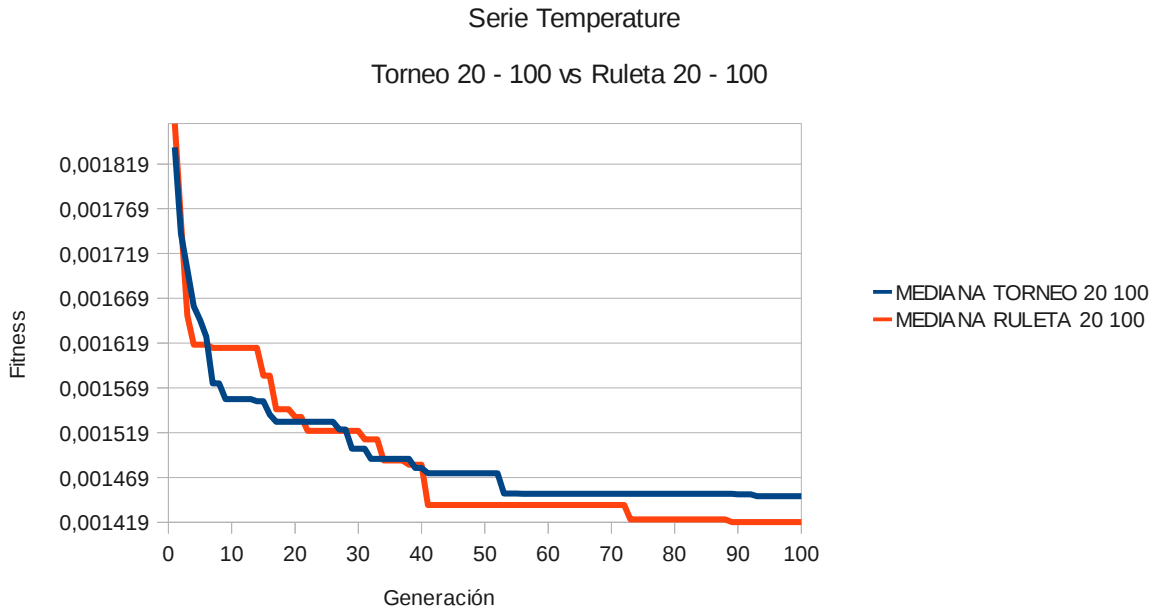


*Figura 37: Gráfica de evolución de la fitness de Temperature. Ruleta 20 individuos.*

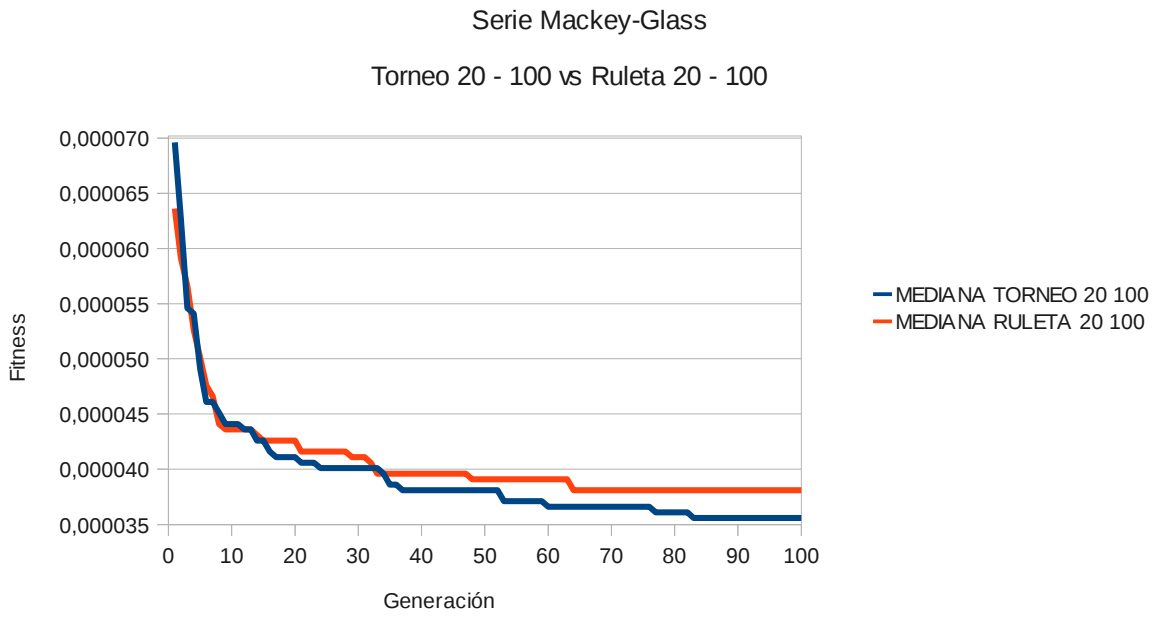


*Figura 38: Gráfica de evolución de la fitness de Mackey-Glass. Ruleta 20 individuos.*

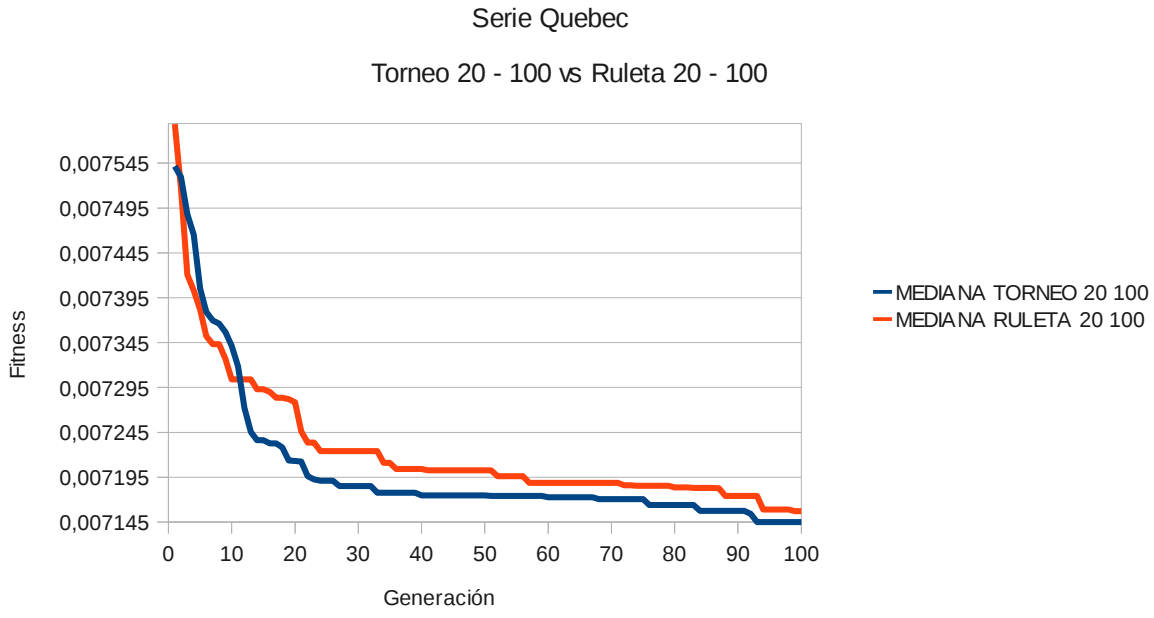


*Figura 39: Gráfica de evolución de la fitness de Quebec. Ruleta 20 individuos.*

Al igual que en las dos últimas experimentaciones, se puede observar una mejora acusada de la evolución en las primeras 30 generaciones en las figuras 36, 38 y 39, y en

las primeras 40 generaciones para la figura 37, serie Temperature, mientras que en la figura 35, serie Dow-Jones, la evolución es proporcional a lo largo de toda la gráfica.

Destacar también que, como en el caso anterior, las curvas de las gráficas de la evolución son muy semejantes a las obtenidas en el experimento de referencia no aportando nada significativo por el hecho de hacer uso del algoritmo de selección de la ruleta.

Resumiendo lo visto, de estás gráficas se puede concluir que los resultados más notables de mejora en la función de evaluación se producen, en la mayoría de ellas, en las primeras 25 generaciones. Destacan las caídas pronunciadas de las líneas de las gráficas en estas primeras generaciones para después reducir notablemente su pendiente.

## 5.4. Resultados SMAPE/MSE

Seguidamente se van a mostrar las tablas que comparan los valores SMAPE y MSE de cada experimento con el experimento base, para ver cual ha sido la mejor. En la primera fila se presenta el valor de SMAPE y en la segunda el valor de MSE. Se resalta en verde el mejor valor de cada experimento.

La tabla 1 muestra los valores obtenidos para el torneo de 2 padres con 50 individuos y 100 generaciones frente al torneo de 2 padres con 20 individuos y 100 generaciones.

| SERIE | ERROR | 100 TORNEO 20 INDIVIDUOS | 100 TORNEO 50 INDIVIDUOS |
|---|---|---|---|
| PASSENGERS | SMAPE | 14,648 | 15,489 |
| | MSE | 0,008805 | 0,009889 |
| TEMPERATURE | SMAPE | 29,484 | 30,666 |
| | MSE | 0,139875 | 0,149166 |
| DOWJONES | SMAPE | 9,33 | 9,575 |
| | MSE | 0,039847 | 0,042892 |
| QUEBEC | SMAPE | 26,305 | 18,728 |
| | MSE | 0,085159 | 0,04639 |
| MACKEY | SMAPE | 5,909 | 5,432 |
| | MSE | 0,001564 | 0,001528 |
| MEDIA | SMAPE | 17,1352 | 15,978 |
| | MSE | 0,05505 | 0,049973 |

*Tabla 1: Valores SMAPE/MSE. Torneo 2 padres con 50 individuos.*

Por lo que se puede observar a simple vista en los resultados, usar menos individuos para series cortas da mejor resultado que usar más individuos, aunque los resultados obtenidos para los errores difieren muy poco en ambos experimentos, sin embargo el tiempo de cómputo es mucho menor al tener menos individuos. En las series largas ocurre lo contrario, es mucho mejor el uso de un mayor número de individuos. En la serie de Quebec se muestra una diferencia notable en ambos errores mientras que en la serie de Mackey-Glass esta diferencia es mucho menos apreciable.

Ahora se muestran los resultados para el experimento de torneo de 2 padres con 20 individuos y 250 generaciones frente al torneo de 2 padres con 20 individuos y 100 generaciones, tabla 2.

| SERIE | ERROR | 100 TORNEO 20 INDIVIDUOS | 250 TORNEO 20 INDIVIDUOS |
|---|---|---|---|
| PASSENGERS | SMAPE | 14,648 | 14,243 |
| | MSE | 0,008805 | 0,008346 |
| TEMPERATURE | SMAPE | 29,484 | 29,696 |
| | MSE | 0,139875 | 0,141749 |
| DOWJONES | SMAPE | 9,33 | 10,303 |
| | MSE | 0,039847 | 0,04991 |
| QUEBEC | SMAPE | 26,305 | 30,131 |
| | MSE | 0,085159 | 0,104937 |
| MACKEY | SMAPE | 5,909 | 7,171 |
| | MSE | 0,001564 | 0,002462 |
| MEDIA | SMAPE | 17,1352 | 18,3088 |
| | MSE | 0,05505 | 0,0614808 |

*Tabla 2: Valores SMAPE/MSE. Torneo 2 padres con 20 individuos, 250 generaciones.*

Según los valores mostrados, no parece que el hecho de hacer un mayor número de generaciones suponga una mejora, ya que tan sólo en el caso de Passengers resulta este experimento ganador con respecto al base y aún así la diferencia es ínfima. En el resto de los casos resulta perdedor e incluso en algunos casos con una diferencia un tanto notable.

En la tabla 3 se pueden ver los resultados de los experimentos de torneo de 3 padres con 20 individuos y 100 generaciones frente al torneo de 2 padres con 20 individuos y 100 generaciones.

| SERIE | ERROR | 100 TORNEO 20 INDIVIDUOS | 100 TORNEO 3 20 INDIVIDUOS |
|---|---|---|---|
| PASSENGERS | SMAPE | 14,648 | 13,806 |
| | MSE | 0,008805 | 0,007959 |
| TEMPERATURE | SMAPE | 29,484 | 29,874 |
| | MSE | 0,139875 | 0,144518 |
| DOWJONES | SMAPE | 9,33 | 9,353 |
| | MSE | 0,039847 | 0,040094 |
| QUEBEC | SMAPE | 26,305 | 26,075 |
| | MSE | 0,085159 | 0,083825 |
| MACKEY | SMAPE | 5,909 | 5,488 |
| | MSE | 0,001564 | 0,001477 |
| MEDIA | SMAPE | 17,1352 | 16,9192 |
| | MSE | 0,05505 | 0,0555746 |

*Tabla 3: Valores SMAPE/MSE. Torneo de 3 padres con 20 individuos, 100 generaciones.*

En este experimento, dada la poca diferencia que hay en todas las comparaciones

de resultados, y como el coste computacional del algoritmo de selección de torneo de 2 padres es un poco menor que el algoritmo de selección de 3 padres, lleva a descartar el uso de este último.

En los últimos resultados que se van a mostrar en la tabla 4 se puede ver la comparación de valores obtenidos en la ruleta con 20 individuos y 100 generaciones frente al torneo de 2 padres con 20 individuos y 100 generaciones.

| SERIE | ERROR | 100 TORNEO 20 INDIVIDUOS | 100 RULETA 20 INDIVIDUOS |
|---|---|---|---|
| PASSENGERS | SMAPE | 14,648 | 14,682 |
| | MSE | 0,008805 | 0,008808 |
| TEMPERATURE | SMAPE | 29,484 | 29,853 |
| | MSE | 0,139875 | 0,143082 |
| DOWJONES | SMAPE | 9,33 | 13,147 |
| | MSE | 0,039847 | 0,0821 |
| QUEBEC | SMAPE | 26,305 | 30,42 |
| | MSE | 0,085159 | 0,10596 |
| MACKEY | SMAPE | 5,909 | 5,745 |
| | MSE | 0,001564 | 0,001782 |
| MEDIA | SMAPE | 17,1352 | 18,7694 |
| | MSE | 0,05505 | 0,0683464 |

*Tabla 4: Valores SMAPE/MSE. Ruleta con 20 individuos y 100 generaciones.*

Al igual que en el caso anterior los valores obtenidos en la tabla 4 revelan que la diferencia entre los errores no es considerable. Si además se resaltan los resultados de la serie Mackey-Glass, donde la diferencia obtenida es bastante mayor, esto lleva a descartar este algoritmo en usos futuros.

Para concluir, en la tabla 5 se puede ver la comparación de las medias de los valores de SMAPE y MSE obtenidos por cada experimento para poder tener una apreciación global de los resultados obtenidos.

| EXPERIMENTO | SMAPE | MSE |
|---|---|---|
| 100 TORNEO 20 INDIVIDUOS | 11,5221 | 0,028307 |
| 100 TORNEO 50 INDIVIDUOS | 15,978 | 0,049973 |
| 100 TORNEO 3 20 INDIVIDUOS | 16,9192 | 0,0555746 |
| 250 TORNEO 20 INDIVIDUOS | 18,3088 | 0,0614808 |
| 100 RULETA 20 INDIVIDUOS | 27,0622 | 0,097571 |

*Tabla 5: Medias de los valores SMAPE/MSE por experimento.*

Como se puede ver en esta última tabla los valores de la media de los experimentos realizados confirman como vencedor al algoritmo de selección de la ruleta de dos padres con 20 individuos de población y generado 100 veces.

# 6 CONCLUSIONES

En este proyecto se han llevado a cabo cuatro experimentaciones que tratan de dar respuesta a las siguientes preguntas partiendo de la situación, también experimentada, de la selección por torneo de 2 padres con 20 individuos y 100 generaciones:

- ¿Qué ocurre si se amplia el número de individuos que se usan en la población del AG? Torneo de 2 padres con 50 individuos y 100 generaciones.
- ¿Qué pasa si en vez de usar 100 generaciones se amplia su número a 250 generaciones? Torneo de 2 padres con 20 individuos y 250 generaciones.
- ¿Y si en vez del algoritmo de selección de torneo de 2 padres se usase el algoritmo de 3 padres? Torneo de 3 padres con 20 individuos y 100 generaciones.
- Y, por último, ¿y si en vez de estos se usará el algoritmo de selección de la ruleta? Ruleta con 20 individuos y 100 generaciones.

Respondiendo a cada pregunta planteada y tras la ejecución y recopilación de los resultados obtenidos con la experimentación se ha llegada a las siguientes conclusiones:

- El hecho de ampliar el número de individuos de la población del AG no implica necesariamente que los resultados obtenidos para series pequeñas sean mejores, ya que, como se puede observar en las tablas 1, 2 y 3, los resultados obtenidos son peores aunque con poca diferencia pero el coste computacional de obtenerlos no compensa. Sin embargo si se puede destacar que el uso de una población mayor para series de datos de mayor tamaño como Quebec y Mackey-Glass si ofrecen una mejora significativa según el resultado visto en la tabla 4 y más ajustado en la tabla 5, lo que puede llevar a plantearse invertir más tiempo computacional para lograr mejores resultados en este tipo de series temporales.

- Si se aumenta el número de iteraciones efectuadas por el AG se ha comprobado, según las tablas 6, 7, 8, 9 y 10, que las mejoras obtenidas no son nada significativas para series pequeñas como Passengers, Dow-Jones y Temperature, y que para las series más grandes se empeoran los resultados obtenidos. Si a esto se le suma el mayor coste computacional que conlleva el aumento de las generaciones lleva a descartar este planteamiento para futuras experimentaciones.
- Al plantear un nuevo algoritmo de selección como es el torneo de 3 padres, se espera que con la introducción de un nuevo padre se amplié el universo de búsqueda del AG, pero según los resultados obtenidos en las tablas 11, 12, 13, 14 y 15 las mejoras obtenidas no son muy llamativas e incluso en los casos de las tablas 11 y 12 son un poco peores.
- Se ha probado un algoritmo de selección alternativo al torneo, el de la ruleta buscando también ampliar la zona de búsqueda mediante la elección parcialmente aleatoria de los individuos a cruzar, pero tampoco los resultados han sido nada positivos como se ha podido ver en las tablas 16, 17, 18, 19 y 20.

Dadas estas aplicaciones se puede concluir que el mejor camino a seguir en futuros experimentos apunta a un algoritmo de selección de 2 padres con 20 individuos y 100 generaciones, al menos para series del tipo Passengers, Dow-Jones o Temperature. En el caso de series mayores como Quebec y Mackey-Glass, se pueden afinar las predicciones aumentando el número de individuos a 50 sacrificando tiempo de cálculo.

# 7 LÍNEAS FUTURAS

En este punto se van a presentar algunas ideas de posibles enfoques que se pueden dar en trabajos futuros de investigación relacionados con la presente memoria.

En base a los experimentos realizados y en su misma línea se van a proponer una serie de trabajos futuros.

En primer lugar, y siguiendo el experimento del incremento del número de individuos de la población del AG, se podría probar a reducir la población por debajo de los 20 individuos, o lo que es lo mismo probar un algoritmo micro-genético. Un algoritmo micro-genético, es un AG con e una pequeña población que evoluciona durante un pequeño periodo de generaciones. El número de evaluaciones del fitness es proporcional al tamaño de la población, de tal forma que deben buscarse soluciones que aboguen por manejar tamaños de población más pequeños que el AG clásico, sin que esto implique una pérdida de diversidad o de capacidad de exploración en los micro-genéticos Básicamente, en los algoritmos micro-genéticos, cuando se detecta que la tasa de convergencia de la población excede un cierto umbral (alrededor del 95% normalmente), entendiendo como tal el grado de similitud entre todos los cromosomas, se reinicia aleatoriamente toda la población manteniendo únicamente la mejor solución obtenida hasta el momento. En el caso de utilizar codificación binaria, la tasa de convergencia se mide como la relación entre el número de bits en los que coinciden todos los cromosomas y la longitud total del cromosoma.

En los experimentos efectuados se ha probado con 100 y 250 generaciones, también se podrían hacer pruebas con un menor número de generaciones de tal manera que con buenos resultados tardemos menor tiempo en encontrar una solución óptima, se comprueba que se obtiene un buen valor para la función de adecuación. Se podrían probar 50, 40, 30 generaciones y así sucesivamente hasta dar con una proporción

generaciones/error aceptable.

Partiendo del algoritmo de selección de torneo de 2 padres usado en el experimento base, y vistos los mediocres resultados obtenidos con el algoritmo de selección de torneo de 3 padres y de la ruleta, se podría variar este algoritmo y probar otros. Existen muchos otros algoritmos de selección. Unos buscan mejorar la eficiencia computacional, otros el número de veces que los mejores o peores individuos pueden ser seleccionados. Algunos de estos algoritmos, entre otros muchos, con una breve descripción son:

- Muestreo determinístico. Selecciona los individuos más aptos de entre toda la población.
- Escalamiento sigma. Es una técnica ideada para mapear la aptitud original de un individuo con su valor esperado, de manera que el AG sea menos susceptible a la convergencia prematura. La idea es mantener más o menos constante la presión de selección a lo largo del proceso evolutivo.
- Selección por jerarquías. Este método de selección ordena a los individuos en función de su aptitud. De un rango determinado se asocia el límite inferior al peor individuo y el límite superior al mejor. Al resto de los individuos se les asigna un valor mediante interpolación de acuerdo a su posición en el ordenamiento. Los valores obtenidos mantienen el ordenamiento de los individuos con base en su aptitud. Al mismo tiempo, al normalizar con base en la posición, elimina las grandes brechas posibles entre individuos, lo que evita la convergencia prematura .
- Estado uniforme. En esta sólo unos cuantos individuos de la población son reemplazados en cada generación, los menos aptos. Esta técnica resulta útil cuando los miembros de la población resuelven colectivamente, y no de manera individual, un problema.
- Sobrante estocástico. la idea principal es asignar determinísticamente las partes enteras de los valores esperados para cada individuo y luego usar otro esquema

(proporcional) para la parte fraccionaria. El sobrante estocástico reduce los problemas de la ruleta, pero puede causar convergencia prematura al introducir una mayor presión de selección.

Para completar este conjunto de ideas se podría aportar otro enfoque de uso del AG, usar el AG para clasificar características de las RNA en vez de para buscar la mejor RNA. Los AGs parecen adecuados para resolver el problema de selección de características debido a su paralelismo inherente, su búsqueda guiada de las regiones más promisorias, su habilidad para encontrar y mantener múltiples óptimos, y su habilidad para optimizar criterios no derivables.

Se podría usar un AG de nichos para la selección de características para clasificadores neuronales. Los métodos de nichos permiten la formación de subpoblaciones estables de tiras binarias diferentes dentro de un AG, permitiendo encontrar y mantener múltiples óptimos locales. En el método de nichos denominado hacinamiento determinístico (*deterministic crowding*), todos los elementos de la población son apareados aleatoriamente y recombinados. La descendencia resultante sostiene un torneo con su padre más cercano (en términos de distancia de Hamming). Los ganadores se copian a la nueva población para la próxima generación.

Para el problema de selección de características, los individuos de la población se representan como tiras binarias, donde un "0" en la i-ésima posición indica que la i-ésima característica se excluye del subconjunto de características, y un "1" indica que la característica está presente.

# 8 BIBLIOGRAFÍA

[1] J. PERALTA, G. GUTIERREZ, A. SANCHIS, "ADANN: AUTOMATIC DESIGN OF ARTIFICIAL NEURAL NETWORKS". ARC-FEC 2008 (GECCO 2008). ISBN 978-1-60558-131-6.

[2] ANÁLISIS DE SERIES TEMPORALES: UN EJEMPLO DE APLICACIÓN EN ÁMBITOS PSICOLÓGICOS. JARA, MARÍA PILAR Y ROSEL, JESÚS. EDITORIAL: UNIVERSITAT JAUME I. FECHA DE PUBLICACIÓN: 01/12/2002. ISBN: 978-84-8021-388-2.

[3] PROF. DR. ANDREAS ZELL, WSI COMPUTER SCIENCE DEPARTMENT, COMPUTER ARQUITECTURE, SOFTWARE, ARTIFICIAL NEURAL NETWORKS HTTP://WWW-RA.INFORMATIK.UNI-TUEBINGEN.DE/SNNS/.

[4] BLICKLE, T. AND THIELE, L. (1995). A COMPARISON OF SELECTION SCHEMES USED IN GENETIC ALGORITHMS. TECHNICAL REPORT 11, CIMPUTER ENGINEERING AND COMUNNICATION NETWORK LAB (TIK), GLORIASTRASSE 35, 8092 ZURICH, SWITZERLAND.

[5] HYNDMAN, R.J. (N.D.) TIME SERIES DATA LIBRARY, HTTP://WWW.ROBJHYNDMAN.COM/TSDL. ACCESSED ON FEBRUARY 1ST IMPROVE THE FORECAST TO OBTAIN AN ACCURATE SYSTEM. 2010.

[6] G.F. MILLER, P.M. TODD AND S. U. HEDGE. DESIGNING NEURAL NETWORKS USING GENETIC ALGORITHM. 1989.

[7] XIN YAO; ISLAM, M.M.; , "EVOLVING ARTIFICIAL NEURAL NETWORK ENSEMBLES," *COMPUTATIONAL INTELLIGENCE MAGAZINE, IEEE* , VOL.3, NO.1, PP.31-42, FEBRUARY 2008 DOI: 10.1109/MCI.2007.913386 HTTP://IEEEXPLORE.IEEE.ORG/STAMP/STAMP.JSP?TP=&ARNUMBER=4442253&ISNUMBER=4442242

[8] HTTP://ES.PAPERBLOG.COM/PREDECIR-EL-FUTURO-COMO-SE-CUENTA-LA-HISTORIA-427600/

[9] HTTP://ES.WIKIPEDIA.ORG/WIKI/ASTROLOGÍA

[10] HTTP://WWW.RA.CS.UNI-TUEBINGEN.DE/SNNS/

[11] HTTP://WWW-ROCQ.INRIA.FR/~CRUCIANU/YANNS/NEURAP98.PDF

[12] HTTP://GREY.COLORADO.EDU/EMERGENT/INDEX.PHP/COMPARISON_OF_NEURAL_NETWORK_SIMULATORS

[13] HTTP://WWW.DOC.IC.AC.UK/~ND/SURPRISE_96/JOURNAL/VOL1/HMW/ARTICLE1.HTML

[14] HTTP://WWW-FORMAL.STANFORD.EDU/JMC/WHATISAI/NODE1.HTML

[15] HTTP://WWW.ANSWERMATH.COM/GENETIC-ALGORITHMS.HTM

[16] HTTP://REDES6.BLOG.COM.ES/2011/05/25/APLICACIONES-11213181/

[17] HTTP://WWW.LEARNARTIFICIALNEURALNETWORKS.COM/

[18] HTTP://EBOOKBROWSE.COM/REDES-NEURONALES-CON-ALGORITMOS-GEN%C3%A9TICOS-PDF-D84783052

[19] HTTP://WWW.NEURAL-FORECASTING.COM/MLP_NEURAL_NETS.HTM

[20] HTTP://IRRIGATION.RID.GO.TH/RID15/PPN/KNOWLEDGE/

[21] HTTP://WWW.DOC.IC.AC.UK/~ND/SURPRISE_96/JOURNAL/VOL4/CS11/REPORT.HTML

[22] HTTP://COEVOLUCION.NET/INDEX.PHP/COMPONENT/CONTENT/ARTICLE/93-ALGORITMOS-GENETICOS

[23] HTTP://EN.WIKIPEDIA.ORG/WIKI/NEURAL_NETWORK

[24] HTTP://WWW.RENNARD.ORG/ALIFE/ENGLISH/GAVINTRGB.HTML

[25] HTTP://WWW.DOC.IC.AC.UK/~ND/SURPRISE_96/JOURNAL/VOL4/TCW2/REPORT.HTML

[26] HTTP://THE-GEEK.ORG/DOCS/ALGEN/

[27] HTTP://WWW.RO.FERI.UNI-MB.SI/PREDMETI/INT_REG/PREDAVANJA/ENG/3.GENETIC %20ALGORITHM/_18.HTML

[28] HTTP://SABIA.TIC.UDC.ES/MGESTAL/CV/AAGGTUTORIAL/TUTORIALALGORITMOSGENETICOS.PDF

[29] HTTP://WWW.CSC.KTH.SE/~ORRE/SNNS-MANUAL/USERMANUAL/USERMANUAL.HTML

[30] HTTP://WWW.STATISTICALFORECASTING.COM/FORECASTING-METHODS.PHP

[31] HTTP://FEDORAPROJECT.ORG/

[32] HTTP://ES.LIBREOFFICE.ORG/

[33] HTTP://ES.WIKIPEDIA.ORG/WIKI/NETBEANS

[34] HTTP://ES.WIKIPEDIA.ORG/WIKI/JAVA_%28LENGUAJE_DE_PROGRAMACI%C3%B3N%29

# 9 ANEXOS

## *9.1. Manual de la herramienta GetForecastedValues*

Debido a lo tedioso que resulta la extracción de resultados y el elevado número de los mismos se optó por desarrollar una herramienta que facilitará dicha extracción.

*GetForecastedValues* tiene dos propósitos, obtener de manera automática los ficheros "*fitness_ga.res*" con la evolución de la fitness de cada experimento, y probar la RNA ganadora de cada experimento obteniendo los resultados en los ficheros "*<fenotipo>.res*" futuros para poder compararlos con otros ya obtenidos.

*GetForecastedValues* es una herramienta multiplataforma ya que ha sido desarrollado en Java y podrá ser ejecutada en cualquier ordenador que disponga de una máquina virtual de Java.

Para ejecutar la aplicación desde un terminal o línea de comandos se tendrá que escribir lo siguiente:

*java -jar GetForecastedValues.jar Path Folder [Values Nv4f]*

Cada argumento de la aplicación se explica a continuación:

1. `Path`. Fichero en formato ZIP, con los archivos comprimidos generados por el experimento, o bien, ruta hacia la carpeta con varios ficheros ZIP que se analizarán uno a uno, generando los resultados oportunos por cada uno de ellos.
2. `Folder`. Ruta dentro de cada fichero ZIP hacia la carpeta que contiene las generaciones del experimento, por ejemplo: nets, Pruebas/nets.

3. Values. Fichero de texto con los valores de la serie temporal usada en el/los experimentos. Sólo se pueden procesar experimentos relativos a la misma serie temporal no a series diferentes.

4. Nv4f. (del inglés Number of Values For Forecast) Número de valores a predecir en el/los experimentos (nv4f). Es el número de valores futuros que se obtendrán al procesar el experimento.

En el caso de que la aplicación se invoque con sólo los dos primeros argumentos *path* y *folder*, extraerá el fichero "*fitness_ga.res*" del fichero ZIP del experimento o de cada fichero ZIP contenido en la ruta al directorio actual.

## *9.2. Código fuente de Batchman de SNNS "batch_2_forecast.bat"*

```
## -----------------------------------------------------------------------------

#### FILE that content a PROGRAM FOR BATCHMAN IN BATCHMAN LANGUAGE

## -----------------------------------------------------------------------------

## -----------------------------------------------------------------------------

## Links for manual on line

##

## Version 4.1

## http://www.lix.polytechnique.fr/~liberti/public/computing/neural/snns/UserManual/node1.html


 print("### INICIO PRIMERA PARTE")

#############################################################################################
##                                                                                         ##

##                  SECTION OF VARIABLES = PARAMETERS OF LEARNING                          ##

## ------------------------------------------------------------------------------------------


## Variables in uppercase are VARIABLE SYSTEMS

## Variables in lowercase are VARIABLE SYSTEMS


# Description file (in SNNS format) of trained net used to forecast values

net_file="Mackey_50_250_10/Mackey_50_250_10/Nets/11_165_91_1562142939_mte_net_file.net"


# Patterns files

pattern_file="Mackey_50_250_10/Mackey_50_250_10/Patterns/11_165_91_1562142939_sv.pat"


# Result patterns file

result_pattern_file="Mackey_50_250_10/Mackey_50_250_10/Patterns/11_165_91_1562142939_sv.pat.res"


# Function for initialitate Weights

func_init_weight="Randomize_Weights"
```

```
# Function for Learning Algorithm

func_learn_alg="Std_Backpropagation"

# Learning rate

alfa=0

# Net_topology

net_topology="Mackey_50_250_10/Mackey_50_250_10/Nets/11_165_91_1562142939_mte_net_file.net"


# Function for Update Weights

func_update_weight="Topological_Order"


# Value to Shuffle patterns

shuf_patterns=TRUE

## ------------------------------------------------------------------------

#### FILE that content a PROGRAM FOR BATCHMAN IN BATCHMAN LANGUAGE

## ------------------------------------------------------------------------

## ------------------------------------------------------------------------

## Links for manual on line

##

## Version 4.1

## http://www.lix.polytechnique.fr/~liberti/public/computing/neural/snns/UserManual/node1.html




##############################################################################
##                                                          ##
##   SECTION OF INSTRUCCTION FOR TEST                                  ##
##                                                          ##
##                                                          ##
##----------------------------------------------------------------------------##


print("### Test to obtain forecast values")

# ------------------------------------------------
```

```
# LOAD NET
loadNet(net_file)

# ----------------------------------------------------
# LOAD PATTERNS
# Put off until Learning process

# ----------------------------------------------------
# Set Parameters of ANN + Learning process

### setLearnFunc( "Std_Backpropagation", 0.1)
setLearnFunc( func_learn_alg, alfa)

### setUpdateFunc ("Topological_Order")
setUpdateFunc (func_update_weight)

# Activation Functions for nodes is determined into network file
# ----------------------------------------------------

# LOAD PATTERNS
loadPattern(pattern_file)
print("Load Patterns ")

setPattern(pattern_file)
print("Set Patterns ")
num_patterns:=PAT
print("num_patterns:",num_patterns)

setShuffle(TRUE) ## To test this parameter could be set to FALSE

## ------------------------------------------------------------------------
```

```
print ("## ----------------------------------")
print (" func_learn_alg", func_learn_alg)
print (" alfa ", alfa)
print ("## ----------------------------------")
print (" net_topology ", net_topology)

print ("PATTERNS:", num_patters)

testNet()
print("testNet")
test_sse:=SSE
test_mse:=MSE
test_ssepu:=SSEPU

print (" test_sse ",test_sse," test_mse ",test_mse," test_ssepu: ",test_ssepu )

## Save the results of train patterns

##setLearnFunc( "Std_Backpropagation", 0)
##trainNet()
saveResult(result_pattern_file, 1, PAT, TRUE, FALSE, "create")
print("#  - save_the_pattern_result: ",result_pattern_file)

print("#### END ")

echo "shuf_patterns=TRUE"
```